MODELING, OPTIMIZATION AND TESTING FOR ANALOG/MIXED-SIGNAL CIRCUITS IN DEEPLY SCALED CMOS TECHNOLOGIES

A Dissertation

by

GUO YU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2009

Major Subject: Computer Engineering

MODELING, OPTIMIZATION AND TESTING FOR ANALOG/MIXED-SIGNAL CIRCUITS IN DEEPLY SCALED CMOS TECHNOLOGIES

A Dissertation

by

GUO YU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Peng Li
Committee Members,	Gwan Choi
	Edgar Sanchez-Sinencio
	Duncan M. Walker
Head of Department,	Costas N. Georghiades

December 2009

Major Subject: Computer Engineering

ABSTRACT

Modeling, Optimization and Testing for Analog/Mixed-Signal
Circuits in Deeply Scaled CMOS Technologies. (December 2009)
Guo Yu, B.S., Fudan University;
M.S., Delft University of Technology
Chair of Advisory Committee: Dr. Peng Li

As CMOS technologies move to sub-100nm regions, the design and verification for analog/mixed-signal circuits become more and more difficult due to the problems including the decrease of transconductance, severe gate leakage and profound mismatches. The increasing manufacturing-induced process variations and their impacts on circuit performances make the already complex circuit design even more sophisticated in the deeply scaled CMOS technologies. Given these barriers, efforts are needed to ensure the circuits are robust and optimized with consideration of parametric variations. This research presents innovative computer-aided design approaches to address three such problems: (1) large analog/mixed-signal performance modeling under process variations, (2) yield-aware optimization for complex analog/mixed-signal systems and (3) on-chip test scheme development to detect and compensate parametric failures.

The first problem focus on the efficient circuit performance evaluation with consideration of process variations which serves as the baseline for robust analog circuit design. We propose statistical performance modeling methods for two popular types of complex analog/mixed-signal circuits including Sigma-Delta ADCs and charge-pump PLLs. A more general performance modeling is achieved by employing a geostatistics motivated performance model (Kriging model), which is accurate and efficient for capturing stand-alone analog circuit block performances. Based on

the generated block-level performance models, we can solve the more challenging problem of yield-aware system optimization for large analog/mixed-signal systems. Multi-yield pareto fronts are utilized in the hierarchical optimization framework so that the statistical optimal solutions can be achieved efficiently for the systems. We further look into on-chip design-for-test (DFT) circuits in analog systems and solve the problems of linearity test in ADCs and DFT scheme optimization in charge-pump PLLs. Finally a design example of digital intensive PLL is presented to illustrate the practical applications of the modeling, optimization and testing approaches for large analog/mixed-signal systems.

To My father, YU Chenghua, and my mother, SHEN Meiqin

ACKNOWLEDGMENTS

I would like to express my great thanks to my advisor, Dr. Peng Li, for his kind guidance for my Ph.D. study. Dr. Peng Li shared his deep knowledge, research experience and insight with me constantly and provided encouragement and support to me during my research work. This dissertation would never have been completed without his advice and help.

I am very grateful for having an exceptional doctoral committee and wish to thank Dr. Edgar Sanchez-Sinencio, Dr. Gwan Choi and Dr. Duncan M. Walker for their invaluable support and advice.

I really appreciate Dr. Hongzhou Liu and Dr. Hui Zhang in Cadence Design System for being my mentors when I was interning there. They shared a great industry experience and insights, which are an excellent treasure to me. In addition, I would like to thank all my friends who made my stay at Texas A&M very enjoyable. In particular I'd like to thank Zhuo Feng, Wei Dong and Xiaoji Ye in Dr. Li's lab for the collaboration and fruitful discussion. Thanks also go to Rajesh Garg, Shiyan Hu, Zhanyuan Jiang in the Computer Engineering Group, and Heng Zhang, Xi Chen and Yung-Chung Lo in the Analog & Mixed Signal Group for their valuable input on my research projects.

My research work was supported by the FCRP Focus Center for Circuit & System Solutions (C2S2), under contract 2003-CT-888. I thank the sponsor for providing financial support.

Last, but not least, I would like to express my greatest gratefulness to my family for their long-lasting encouragement and support.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

As CMOS technologies move into nano-scale regions, the impact of manufacturing-induced variations becomes more and more profound to integrated circuit performances. The continuous feature size scaling causes increasing uncertainties in device and circuit electrical characteristics, as illustrated in Table I. Consequently, circuit performances are no longer deterministic values and the downgraded statistical performances result in parametric failures which in turn cut down the yield of fabricated chips [2]. Therefore the effects of process variations must be taken into consideration in the circuit design stage so that the circuits can work properly once they are fabricated. This requirement, however, is nontrivial and needs significant efforts in understanding device level characterizations and utilizing these information to achieve robust circuit design.

Table I. ITRS predicted process variabilities [1].

Year	08	09	10	11	12	13
Pitch(nm)	57	50	45	40	36	32
V_{th}	37%	42%	42%	42%	58%	58%
Delay	46%	49%	50%	53%	54%	57%
Power	57%	57%	58%	58%	59%	59%

On the other hand, the proliferation of communication and consumer electronic systems leads to high demands for low-power & high-performance analog/mixed-signal circuits, either as stand-alone components or integrated IPs [3]. The design

This dissertation follows the style of *IEEE Transactions on Automatic Control*.

difficulties and complexities of modern analog/mixed-signal circuits have raised significantly in recent years to accomplish sophisticated demands like multiple wireless standards and ultra low power applications. The circuit design tasks become even more challenging when process variations come into play. Statistical performance analysis and optimization requirements in addition to the already lengthy analog and mixed-signal system designs are pushing the product time-to-market to some far ends which deeply hampers product profitability. Innovations are needed to address these challenges.

In this dissertation, we focus on addressing the problems related to robust analog/mixed-signal circuit design in modern CMOS technologies using computer-aided approaches. The most fundamental questions to be answered is: what the circuits will behave after fabrication when so many process uncertainties play roles. Once we have the knowledge about the manners how the circuits behave, it is possible to use these information to enhance circuit designs. Clearly there could be various approaches to achieve system enhancements. One way is to consider the possible performance downgrades in the early design stage and find design solutions that can generate best overall system performances after fabrication. Other possible solutions take advantage of the cheap digital processing capabilities in CMOS technologies. We can develop built-in circuitries to detect performance failures due to the process variations and use digital control logic to compensate performance collapse in individual chips. In the rest of this dissertation we explain these ideas and how to achieve the goals in detail.

A. Capture Statistical Performances Under Process Variations

Although the hardware computation power has been climbing for decades, direct use of SPICE-like simulators to evaluate statistical system performances are still prohibitively expensive for complex analog/mixed-signal systems, especially for the types of circuits with oversampling (e.g. Sigma-Delta ADCs and DACs) or coexistence of slow and fast signals (e.g. Phase-locked Loops) [4, 5]. In these systems, a single run of performance evaluation might take a few days or even weeks using dedicated transistor-level simulators on today's most powerful servers. Therefore, if we target to address the problems of robust analog/mixed-signal design under process variations, it is crucial to find efficient ways to measure system performances.

There is no general solution yet to possess both accuracy and efficiency for complex analog/mixed-signal circuit simulation. In order to speed up circuit evaluation procedure, some noncritical information have to be discarded to trade for simulation speed. This treatment, however, is very circuit topology dependent since different kinds of circuits operate in various manners. The system performance analysis become even more challenging for scaled technologies when process variations have to be counted and the system performances are now statistical variables. The accurate capturing of system performance variations in terms of uncertainties in the device level need to be developed to safeguard circuit design.

Our first attempt to solve this problem is developing efficient statistical system performance evaluation methods for two popular types of analog/mixed-signal circuits including Sigma-Delta ADCs and charge-pump PLLs. Parametric look-up tables are utilized to capture the performance variations of the integrators, quantizers and feedback DACs caused by process uncertainties. Efficient statistical performance evaluation can be achieved with four orders of magnitude runtime speedup over SPICE-like

simulators, which makes robust Sigma-Delta ADC design possible. We use a different approach in PLL system simulation to acknowledge the circuit uniqueness. Efficient behavioral simulation framework is developed to map building block performances to the system level very efficiently. For block performance variation analysis, we propose an efficient parameter-reduction modeling technique to encode process variations into block performance models without much computation effort. In such way we achieve the accurate and efficient PLL performance evaluation taking consideration of process variations.

A further step to solve the statistical performance analysis problem leads to developing more general modeling method. It is well known that the performances of complex analog/mixed-signal systems are very nonlinear and difficult to be modeled accurately. However, for smaller scale analog circuits the performance-design variable relationship are much simpler and can possibly be linked together using elaborated mathematical formulation. We employ Kriging models [6] to capture circuit performances in terms of design parameters and process variables, and utilize these models to evaluate circuit performances statistically. One of the unique characteristics of Kriging performance model is that it can evaluate performance prediction uncertainty for the new input variable set, which make it possible to enhance model accuracy adaptively. This circuit performance modeling method can not only be used to evaluate the performances of circuit blocks, but can also be utilized to achieve more challenging task of complicated system performance optimization.

B. Automatic Yield-aware System Synthesis

The research and development of automatic analog design have been around for decades and are drawing more attention recently as system complexities continue

to grow [3]. Analog design automation methodologies have been implemented as various design-assist tools commercially available. However, most of these tools are limited in terms of design variable number (e.g. can only handle simple amplifier or oscillator designs). A more challenging problem in these tools is the lack of capability to handle process uncertainness in automatic circuit designs for yield safeguard. The gap between the demands of advanced analog designs and offering of current automation tools must be filled.

We propose to utilize the powerful performance modeling capabilities provided by Kriging models for automatic analog system design. Since the mapping from the design and process variables to the circuit performances can be accurately captured by Kriging modeling approach, we can search in the design space directly using the Kriging performance models. In order to achieve this goal, we first sample in the whole design space uniformly to find the optimum design points in global sense. Then we start from the initial design solutions to perform local fine search for better design solutions. The achieved new circuit performance trade-offs are compared with the previous ones unless they have reached convergence. The device-level uncertainties are considered during the optimization search. We use the best achievable performances at the required yield levels as the optimization objects, so the design points obtained in the optimization can guarantee to achieve the required yield levels. A novel simulation algorithm is proposed to help statistical performance distribution analysis, which save about 60% to 70% computation time when compared with the brute-force use of Monte-Carlo simulation.

Performance modeling becomes more complicated for large analog/mixed-signal systems and even Kriging model may not work properly when the performance mapping gets over nonlinear. The other barrier for the direct application of Kriging models in large analog system performance modeling is the model complexity. As

the number of input variables goes up, even the linear increase of Kriging samples leads to quadratic computation cost increase, which prevents us from directly employing the performance models in large system optimization. To solve this problem, we adopt the hierarchical optimization idea. The complex systems are first decomposed into several well-isolated building blocks of smaller sizes, then optimization searches are carried out individually for these blocks. The most difficult problems in this framework are how to preserve process variation information in the final system-level performance evaluations and how to handle the interactions between different building blocks. We tackle these challenges by introducing the concept of multi-yield pareto fronts to capture "near-optimal" design solutions in each building block and perform statistical performance analysis at system level by doing Monte-Carlo simulation for each transistor-level device. In such way we can achieve efficient and accurate yield-aware design synthesis for large analog/mixed-signal systems.

C. Enhance Performance Using On-chip Design-for-test Function

Besides performing the yield-aware design concept to safeguard system robustness, it is also possible to make use of the powerful digital processing capabilities in advanced CMOS technologies to fight against process variations. The immediate requirement in this framework is to detect performance failures induced by process variations, then we can either carry out performance compensations or screen out the faulty chips. In this dissertation we will focus more on the circuit failure detection side including on-chip design-for-test (DFT) circuit design and optimization.

In commercial integrated circuit production, it is not cost-efficient to use sophisticated external measurement instruments or stations to test individual chips. Some circuit performances, for example, linearities of ADCs and lock-in time of PLLs can

not be measured by simple logic on chip. So the idea of self-test relies on applying some easy to measure circuit metrics to test these non-accessible system performances. In order to validate these indirect performance measurement, the metrics selected need to be highly correlated to these hard-to-measure target performances, also we need to know the exact mapping relationship between the performance indicators and the actual performances. We start from developing a cheap on-chip linearity test scheme for Sigma-Delta ADCs with easy-to-access frequency distortion metrics. Since direct linearity tests require super linear reference signal and mass data processing, it is of advantage to replace it with frequency domain measurement. In order to achieve this, we conduct discrete-time Volterra series analysis in the frequency domain and build correlation model between linearities and distortions. Experimental results show excellent performance testing accuracy for the proposed method.

The cost of self-test circuit implementation is very critical in the whole system design. The overhead needs to be kept in minimal while achieving the required test accuracy. So it is crucial to optimize the DFT circuit schemes so that the system can achieve the best overall performances. We employ the simulation framework developed in Chapter II to facilitate the DFT circuit optimization for charge-pump PLLs. Several potential self-test schemes are proposed to detect the parametric failures caused by process variations. Novel circuit-level macromodeling and powerful statistical dimension reduction techniques are employed to evaluate the effectiveness of each on-chip test scheme. We further perform optimization for the chosen scheme topology by configuring internal scheme variables to achieve the optimal balance of implementation cost and test accuracy.

D. Design Case of All-digital PLL

We demonstrate the usefulness of modeling, optimization and testing ideas using a practical design example. The design case selected is a digital intensive PLL, a.k.a. all-digital PLL (ADPLL) [7]. It is different from most of the conventional types of PLL since all the control signals are in the digital domain. ADPLL, however, is still analog/mixed-signal circuit since it relies on the analog circuit blocks like digital controlled oscillator (DCO) to generate clock signals of adjustable frequencies and time-to-digital converter (TDC) to measure frequency and phase differences. ADPLLs are more process variation resistive than most of other mixed-signal circuits thanks to the digital-intensive implementation, although the analog blocks are still affected by the device-level uncertainties.

The uniqueness of ADPLL operation needs to be reflected in the system design and optimization procedures. The most straightforward change is the discretized design values in system implementation, which need to be treated differently from analog-type variables. Typical implementations of ADPLLs require more transistors than the analog counterparts, so it is important to develop an efficient system performance simulation approach to evaluate the design solutions. The digital signal processing is robust to process variations, so the yield-aware optimization framework need to be modified to account for the circuit changes. We develop efficient variation-aware performance models for building blocks and employ a two-step optimization methodology including topology selection and block fine tuning to achieve the efficient ADPLL system optimization.

Self healing of system performances are easier for ADPLLs thanks to the digital type implementation. Many hard-to-detect circuit behaviors can now be measured efficiently with digital signal processing. For example, PLL lock-in time can be obtained

using simple digital logics to monitor digital frequency control signal. Other circuit performances, however, still need elaborate measurement planning. We develop an approach to capture the correlation between the system jitter performances and the frequency differences so that the jitters in ADPLLs can be measured on chip. With the system performance information measure by the DFT circuits, we can implement the performance compensation functions to achieve circuit self healing. The digital implementation of ADPLL systems give us the freedom to adjust filter characteristics and TDC/DCO configurations without much hardware overhead. A prototype of adaptive performance compensation scheme is implemented by measuring the frequency shift levels which represent the ADPLL jitter performances, a throughout search of different system configurations is trigged if the jitter exceeds the predefined value. The configuration with best system performances is saved after the search and the ADPLL is configured to this structure with enhanced system performances.

CHAPTER II

CIRCUIT PERFORMANCE MODELING UNDER PROCESS VARIATIONS *

In this chapter we focus on developing circuit modeling and simulation techniques to achieve efficient and accurate system performance evaluation under consideration of process variations. First we present efficient modeling techniques for two popular mixed-signal circuits including Sigma-Delta ADCs [8] and charge-pump PLLs [9], then we propose a more general circuit performance modeling approach employing Kriging models [10].

A. Lookup Table Based Sigma-Delta ADC Modeling

Sigma-Delta ($\Sigma\Delta$) ADCs have been widely used in data conversion applications due to the good performances. However, oversampling and complex circuit behaviors render the transistor-level analysis of these designs prohibitively time consuming. The inefficiency of the standard simulation approach also rules out the possibility of analyzing the impacts of a multitude of environmental and process variations critical in modern VLSI technologies. We present a lookup table (LUT) based modeling technique to facilitate much more efficient performance analysis of $\Sigma\Delta$ ADCs. Various transistor-level circuit nonidealities are systematically characterized at the building block level

* Part of this chapter is reprinted, with permission, from "Efficient Lookup Table Based Modeling for Robust Design of Sigma-Delta ADCs", by G. Yu and P. Li, *IEEE Trans. on Circuits and Systems - I*, vol. 54, No. 7, pp. 1513-1528, July 2007.

and the whole system is simulated much more efficiently using these building block models. Our approach can provide up to four orders of magnitude runtime speedup over SPICE-like simulators, hence significantly shortening the CPU time required for evaluating system performances such as SNDR (signal-to-noise-and-distortion-ratio). The proposed modeling technique is further extended to enable scalable performance variation analysis of complex ΣΔ ADC designs.

1. Sigma-Delta ADC Background

As illustrated in Fig. 1, the two basic components of ΣΔ ADCs are modulators and digital filters. The analog input is sampled by a very high frequency clock in the ΣΔ modulator, then the signal is passed through a loop-filter to perform noise-shaping. The output of the loop-filter is quantized by an internal A/D converter, producing a bit-stream at the same speed as the sampling clock. A low-pass digital filter then removes the out-of-band noise and the down-sampler converts the high speed bit-stream to the high resolution digital codes.

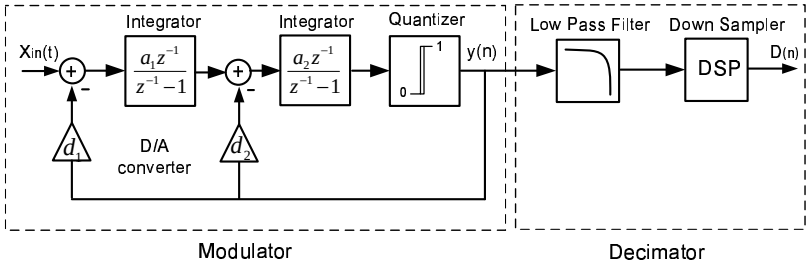

Fig. 1. Block diagram of ΣΔ ADC.

The major components of a ΣΔ modulator are integrators, internal quantizers and D/A converters. The whole system is clocked by an external sampling clock, which makes it possible to model the performance of each component at sampling

intervals. The output of switched-capacitor integrators used in the $\Sigma\Delta$ converter is a function of input signals and their previous states

$$y[k+1] = F(y[k], x[k+1], d[k+1]), \tag{2.1}$$

where $y[k+1]$ is the current output of the integrator, $y[k]$ is the previous integrator output, F is a nonlinear function describing the state transfer, $x[k+1]$ and $d[k+1]$ are the current input signal and feedback digital output, respectively. As shown in Fig. 2, since each integrator is clocked by the sampling clock, it is possible to use lookup tables to model the output at the end of each clock cycle, as described in the later sections.

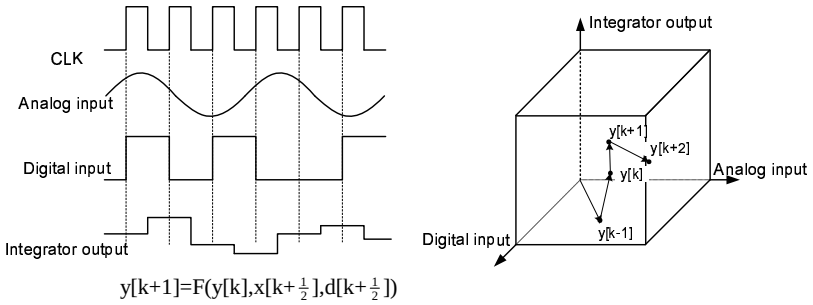

Fig. 2. Clocked $\Sigma\Delta$ modulator behavior.

On the other hand, various circuit-level nonidealities such as the finite DC gain, bandwidth and slew rate of the operational amplifies, charge injections of the switches, mismatching of the internal quantizers and D/A converters, etc, are difficult to analyze accurately by hand analysis, neither are their impacts on system performances. The finite bandwidth, slew rate and saturation of the amplifier also introduce incomplete charge transfer, which shifts system transfer function. There is no simple way to calculate the influence of the effects mentioned above in terms of SNDR, so normally transistor-level simulation needs to be employed. Additionally, using transistor-level simulation to predict the linearity of the design can be prohibitively expensive. For

example, transient analysis needs to be performed over at least $(2^{14}-1)\cdot 128$ clock cycles to fully characterize a $\Sigma\Delta$ ADC. The process variation analysis is another challenge for the conventional simulators because a large number of long transient simulations are needed to evaluate the performance of the circuit at different parameter corners. In the following sections, it will be shown that these issues can be well addressed by adopting LUT based modeling.

2. Look-up Table Modeling

The proposed lookup table (LUT) based simulation framework is illustrated in Fig. 3. In our fast ADC simulation methodology, various circuit blocks are modeled as follows. The macromodel of each building block of the $\Sigma\Delta$ modulator is extracted at the transistor-level using Cadence Spectre [11]. SNDR and THD are calculated using Fast Fourier transform (FFT) to estimate the performance of the modulator. A two-stage Cascaded Integrator Comb (CIC) filter is implemented as the decimator and the decimation rate (same as OSR) is programmable to adjust for different applications. Since the simulator can run long transient simulation very efficiently (2 seconds for 64K cycles), INL and Differential Nonlinearity (DNL) can be easily calculated by applying an input ramp signal and evaluating the output digital codes.

a. Details of Model Extraction Setup

The setup of $\Sigma\Delta$ ADCs for the lookup table generation can be divided into two parts, one consists of integrators and D/A converters, and the other consists of the quantizer. The integrator output is a function of the input signals and the initial state of the integrator, which is discretized to generate the lookup table. The number of levels depends on the accuracy requirement of the simulator. Since $\Sigma\Delta$ ADCs are quite linear in most cases [4], linear interpolation is good enough. The internal

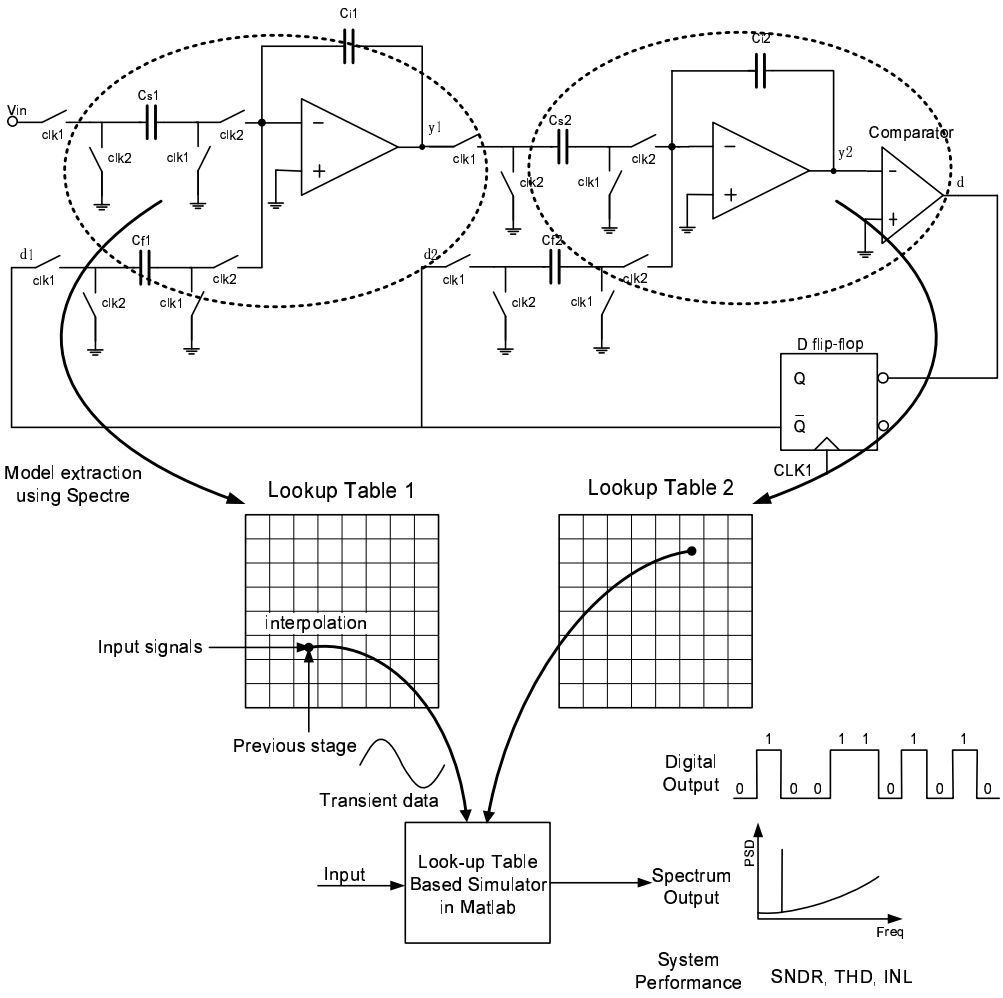

Fig. 3. The proposed LUT based simulation framework.

voltage swing is determined by the system architecture, for low-voltage designs the internal voltages can change from 0 to V_{dd}. To cover the whole range of the voltage swing, we discretize the inputs and outputs of the integrators at N levels, from 0 to V_{dd}.

The extraction setup for an integrator with a 2-bit DAC which is implemented in thermometer code is shown in Fig. 4. A large inductor L together with a voltage source V_s is used to set the initial value of the integrator output. The input of the integrator is also set by a voltage source V_i. The digital output of the quantizer controls the amount of feed back charge. The digital codes of 00 to 11 can be represented by counting the number of three voltage sources V_{d1}, V_{d2} and V_{d3} which are set to V_{dd}. For the $\Sigma\Delta$ modulators with 1-bit quantizer, the modeling of the digital signal is simpler, with only one voltage source V_d.

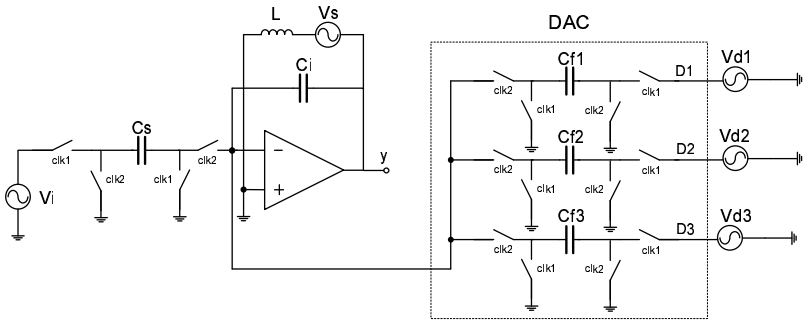

Fig. 4. Model extraction setup for integrators.

The nonidealities of the quantizer should also be taken into consideration. For a 1-bit quantizer, we can use Spectre to find the voltage levels where the digital output switches from 0 to 1 (V_{off+}) and 1 to 0 (V_{off-}), the quantizer is then modeled as

$$d[k+1] = \begin{cases} 1 & (V_{in}[k+1] > V_{off+}) \\ d[k] & (V_{off-} < V_{in}[k+1] < V_{off+}) \\ 0 & (V_{in}[k+1] < V_{off-}) \end{cases} \tag{2.2}$$

where $d[k+1]$ is the current output of the quantizer, $d[k]$ is the output of the quantizer in the previous clock cycle. A 2-bit quantizer can be modeled in a similar way since it is built from three 1-bit quantizers each of which is modeled as in Eqn. 2.2.

b. Controlling of Model Accuracy

To achieve good accuracy for the LUT methodology, several issues regarding with modeling must be taken into consideration. When we perform the lookup table generation, the inductor L and the voltage source V_s are used to set the initial condition of the integrator as shown in Fig. 5, here we redraw it in Fig. 5.

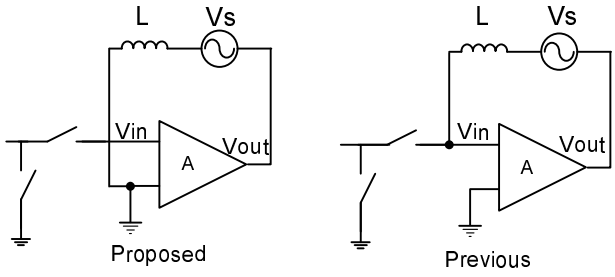

Fig. 5. Modified output voltage setup.

As shown in the right part of Figure 5, the initial output voltage of the integrator was not setup correctly in [12]. To see this, suppose that the gain of the amplifier is A, the input voltage is V_{in} and the initial voltage of the amplifier output is V_s, we get

$$A \cdot V_{in} = -(V_s - V_{in}), \tag{2.3}$$

so the voltage at the input node can be written as $V_{in} = -V_s/(1 + A)$. Suppose $V_s = V_{dd}/2$ and $A = 60dB$, we will get an offset voltage of $|V_{in}| = V_{dd}/2002$. This offset voltage occurs each time when the table is built and will be transferred to the amplifier output through V_s, which makes the tables generated inaccurate. In our experiments, it has been observed that such an offset voltage can introduce as much

as 5dB error in SNDR with the input signal in full voltage swing.

Another issue to be noticed is that, if we take a look into the charging consequence of the integrator in one clock cycle from nT to $(n+1)T$, we can see that the sample-and-hold circuit follows the input signal during the first half clock cycle. So we have to use the input signal at $(n+1/2)T$ instead of $(n+1)T$ as the index to the LUT during simulation. This should be taken into consideration and the Eqn. 2.1 will be rewritten as $y[k+1] = F(y[k], x[k+1/2], d[k+1/2])$, which was not handled correctly in [12]. Since the digital output of the DAC remains the same within a clock cycle, $d[k]$ can be used to replace $d[k+1/2]$.

3. Parametric LUT-based Macromodeling

Environmental and process variations can introduce noticeable shifts in the performance of $\Sigma\Delta$ ADCs. To allow a feasible variation analysis, we combine the response surface modeling with the LUT based methodology in this section. The term circuit design variable includes environmental variation, process variation, mismatching, etc, which has an impact on the circuit performance.

a. Response Surface Modeling

To find the influence of different circuit parameters to the performance of a system, parameterized lookup tables can be used to approximate the system performance under the circuit parameter variations as illustrated in Fig. 6. Given a set of n responses $y_1, y_2, \cdots, y_n$ and n sets of m input variables $x_1, x_2, \cdots, x_m$ which represent the environmental and process variations, we can determine a set of simplified formulas

$\hat{h}_1, \hat{h}_2, \cdots, \hat{h}_n$ to relate x and y as [13]

$$\begin{aligned} \hat{y}_1 &= \hat{h_1}(x_{11}, x_{12}, \cdots, x_{1m}) \\ \hat{y}_2 &= \hat{h_2}(x_{21}, x_{22}, \cdots, x_{2m}) \\ &\vdots \\ \hat{y}_n &= \hat{h_n}(x_{n1}, x_{n2}, \cdots, x_{nm}) \end{aligned} , \tag{2.4}$$

where $\hat{y}_i$ ith approximated response,

$\hat{h_i}$ a function relating y and x,

x_i ith set of circuit variables,

m number of circuit variables,

n number of experimental runs.

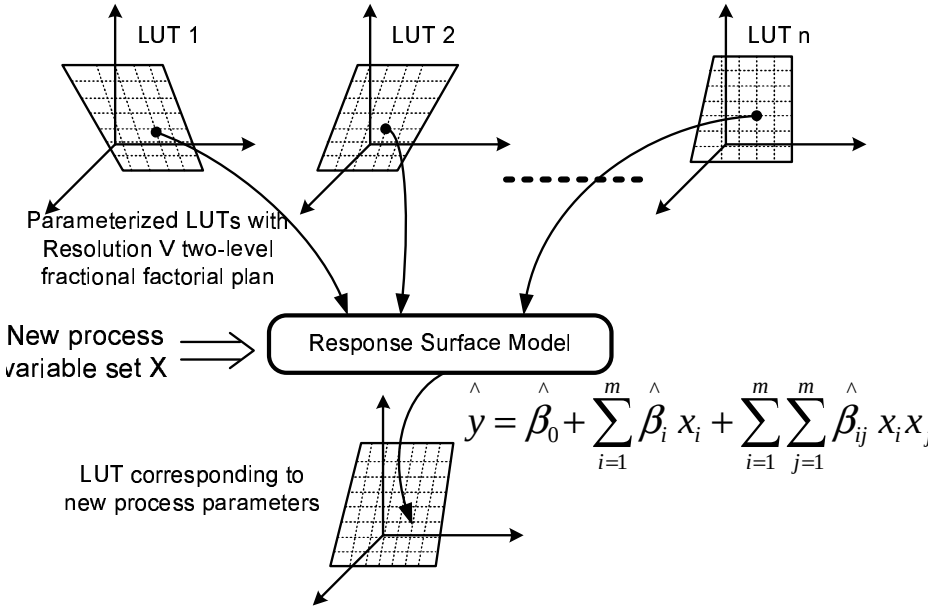

Fig. 6. Response surface modeling of parameterized LUTs.

To construct the parameterized LUTs, a number of simulations should be performed to get enough sets of response values. In order to minimize the cost of LUT generation while keeping reasonable accuracy, RSM can be applied to generate a

quadratic function relating each entry in the table with the circuit parameters [13]

$$\hat{y} = \hat{\beta}_0 + \sum_{i=1}^{m} \hat{\beta}_i x_i + \sum_{i=1}^{m} \sum_{i=1}^{m} \hat{\beta}_{ij} x_i x_j, \tag{2.5}$$

where

x_i ith circuit variable,

$\hat{y}$ approximated response (an table entry),

$\hat{\beta}$ estimated fitting coefficients,

m number of circuit variables.

Since minimizing the number of simulation is a major consideration, an experimental plan to reuse all runs which are performed during variable screening for fitting coefficients construction is needed [13]. Here a second-order central composite plan consists of cube design plan and star design plan is employed [14]. The cube design plan is a two-level fractional factorial plan which can be used to estimate first-order effects (e.g., x_i) and interaction effects (e.g., $x_i x_j$), but it is not possible to estimate pure quadratic terms (e.g., x_i^2). The star design plan is used as a supplementary training set to provide pure quadratic terms in Eqn. 2.5.

When we perform variation analysis, the range for each circuit variable is specified at the very beginning. The discretization of the circuit variables can be arbitrary, but in the two-level factorial experimental design plan, each factor takes on two values -1 and $+1$ to represent the minimum and the maximum value of the circuit variable, respectively. Each factor in the star plan takes on three levels $-\alpha, 0, \alpha$, where 0 represents the circuit variable in nominal case and $\pm\alpha$ are two standardized values of the circuit variable between the two ends $(-1, +1)$ of the circuit variable. Once the experimental plan has been determined by RSM, the fitting coefficients in Eqn. 2.5 can be constructed using least-square fitting. In reality, numerically more stable algorithms such as SVD can be used to solve the nonlinear least square problem.

4. Circuit Examples

We use two second-order ΣΔ ADCs designed with a 1-bit and a 2-bit quantizer as design example to demonstrate the effectiveness of the proposed simulation framework. Both converters are implemented in $0.13\mu m$ CMOS technology with single $1.5\ V$ supply. The oversampling ratio is set to 128 and the sampling clock used is 1MHz and the stimuli is a $2kHz$ $0.8V_{p-p}$ sinusoidal signal. For each converter, we perform 65,536 (2^{16}) + 100 clock cycles transient simulation with the first 100 points thrown. A Kaiser window is applied to the digital output in FFT analysis. We consider process and environment variation for the ΣΔ ADC with 1-bit quantizer, and the internal DAC mismatching for the ΣΔ ADC with 2-bit quantizer. Because the 1-bit DAC in the ΣΔ ADC is quite linear, nonlinearities are mainly due to process variation, also the major part of nonlinearities in the 2-bit ΣΔ ADC come from the mismatching of the internal DACs, this approach is a good approximation for circuit performance evaluation.

The generation of parameterized LUTs is time consuming because traditional simulators are used for transient simulation, while since the tables are reusable once they are built, the speed up of the LUT based simulator is calculated by the ratio of the run time of the LUT based simulator and Spectre. A comparison of the model extraction time, simulation time, SNDR and THD of the LUT based simulator and Spectre are shown in Table II.

Fig. 7 shows the the output spectrum of the ΣΔ ADC with 2-bit quantizer in nominal case. Two spectra fit very well, especially for the signal and distortions which are of most interest in the performance analysis, indicating the good accuracy of the LUT simulator.

The eight parameters selected to represent the environmental and process vari-

Table II. Runtime and accuracy comparison for the proposed simulator.

	LUT based simulator					**Spectre**		
Design	Nom.	Para.	Time	SNDR	THD	Time	SNDR	THD
SDM (1bit)	7 min	9.5 hr	2 s	73.8 dB	-63.1 dB	4.5 hr	74.1 dB	-62.6 dB
SDM (2bit)	20 min	15 hr	4 s	86.8 dB	-76.2 dB	9.5 hr	86.2 dB	-76.8 dB

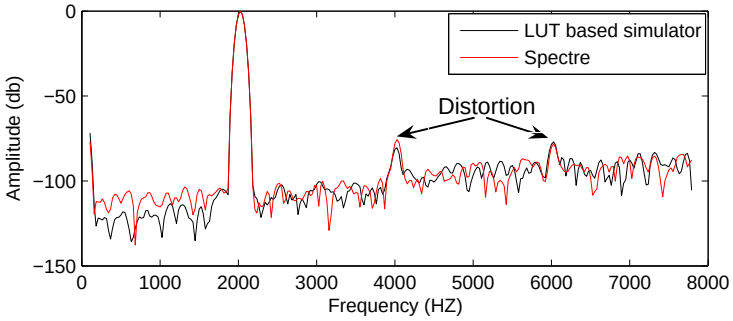

Fig. 7. Spectrum comparison of Spectre and proposed simulator.

ation of the ΣΔ ADC with 1-bit quantizer are presented as follows. Temperature ($Temp$) changes from $-97C^o$ to $127C^o$, the threshold voltage of PMOS (V_{thp}) and NMOS (V_{thn}), the carrier mobility in PMOS (μ_{op}) and NMOS (μ_{on}), the effective channel length of PMOS (L_{effP}) and NMOS (L_{effn}), and the oxide thickness (T_{ox}) are all swept ±40% from their nominal values. A Resolution VI 2^{8-2} fractional factorial design plan that includes 64 runs for the cube design plan and 17 runs for the star design plan is used to build the parameterized lookup table. The high speed of the LUT simulator makes it possible to perform statistical simulation, which is too time consuming to be done with conventional simulators. A statistical performance analysis is carried out with 1000 runs of transient simulation. It will take 4500 hours for Spectre to complete, but only 20min for the LUT simulator. Four sets of environmental and process parameters are selected randomly and transient analysis of 64K

Table III. Comparison of Spectre and LUT based simulator.

	Spectre	LUT simulator	Error	Speed up
set1	72.4dB	72.1dB	0.3dB	8100X
set2	73.2dB	72.3dB	0.9dB	8100X
set3	74.0dB	73.9dB	0.1dB	8100X
set4	74.8dB	74.6dB	0.2dB	8100X

clock cycles are performed to evaluate the accuracy of the LUT simulator, as shown in Table III. The error of SNDR is within 1dB, which demonstrates the effectiveness of our method for capturing process variation.

An experiment of SNDR distribution of the 1-bit quantizer $\Sigma\Delta$ ADC with eight parameters swept randomly for 1000 runs is shown in Fig. 8. We can see that the ADC is most likely to have a SNDR of 73dB and the deviation is small, which means the design of the converter is very robust.

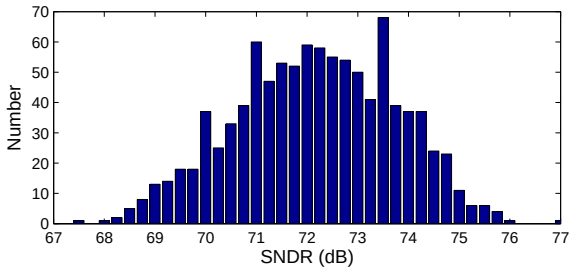

Fig. 8. SNDR distribution with random parameter sweeping.

In today's CMOS technology, the mismatching of capacitors can be controlled within $\pm 1\%$, here we set the maximum mismatching to $\pm 2\%$ to cover all the situations. Statistical simulations are performed to analyze the influence of the mismatching of the two internal DACs in the $\Sigma\Delta$ ADC with 2-bit quantizer by sweeping the capacitances of the three charging capacitors within $\pm 2\%$ randomly for each DAC. The

distributions of SNDR for the Sigma-Delta ADC using two DACs are shown in Fig. 9. We can see from the two figures that the mismatching of the DAC connected to the

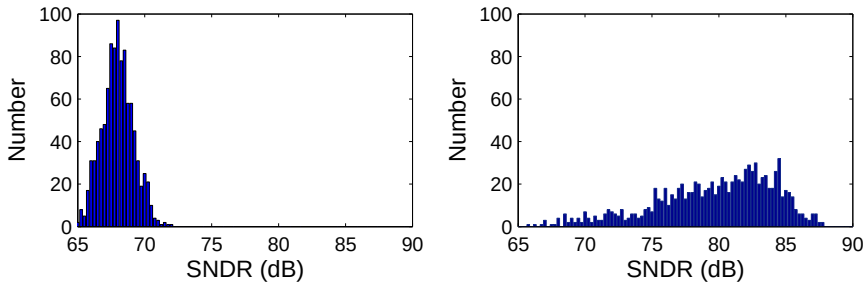

Fig. 9. SNDR distributions with mismatching of DACs.

first stage integrator (left figure) has much more influence to the system performance then that of the other DAC (right figure). This can be explained by the fact that the first DAC is connected directly to the input, so the feedback error because of the DAC mismatching will be magnified by the second stage integrator, which makes the mismatching more severe. This analysis provides useful information to the designer that more attention should be paid to the first stage DAC to make it linear.

B. Parameter Dimension Reduced Phase-Locked Loop Modeling

As an essential building block, PLLs are widely used in today's communication and digital systems for purposes such as frequency synthesis, low-jitter clock generation, data recovery and so on. Although the input and output signals of PLLs are in the digital domain, most PLLs implementations consist of both digital and analog components, which make them prone to process variation influences. In this section we propose an efficient parameter-reduction modeling technique to capture process variations and further achieve low-cost system performance evaluation using hierarchical system simulation. The proposed method can not only be used for robust PLL design

under process variation, but also paves the road for effective built-in self-test circuit design as to be discussed in Chapter IV.

1. PLL Background

As illustrated in Fig. 10, a typical charge-pump PLL system consists of a frequency detector, a charge pump, a loop filter, a voltage-controlled oscillator (VCO) and a frequency divider. The frequency of the output clock signal F_{out} is N times of that of reference clock signal F_{ref}, where N can be an integer number or fractional number. The PLL design options include VCO topologies and component sizes, filter characterizations, charge current in the charge pump and so on. The metrics of PLL systems usually include acquisition/lock-in time, output jitter, system power, total area, etc. Capturing these performances are of great importance in PLL design.

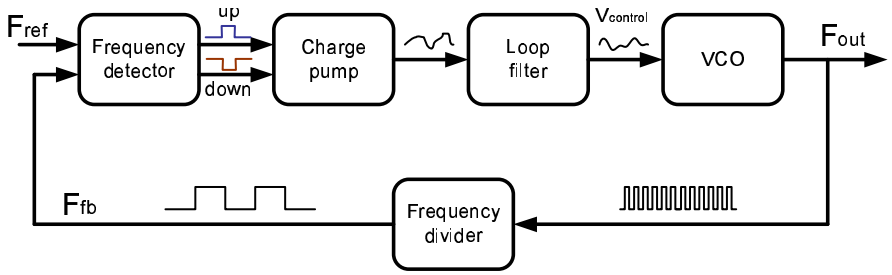

Fig. 10. Block diagram of charge-pump PLL.

Due to the mixed-signal nature, the modeling of PLL system is quite complex and costly. For example, a long transient simulation (in the order of hours or days) is needed to obtain the lock-in time behavior of PLL, which is one of the most important performance metrics for a PLL. The difficulty of system performance evaluation can be addressed by adopting a bottom-up modeling and simulation strategy. The performances of analog building blocks can be evaluated and optimized without too much cost. When the behaviors of analog building blocks are extracted, these building

blocks can be mapped to Verilog-A models for fast system level evaluation [15]. By using this approach we can avoid the scalability issue associated with time consuming transistor-level simulations.

When process variations are considered, the situation becomes more sophisticated. The large number of process variables and the correlations between different building blocks introduce more uncertainties for PLL performance under process variations. In order to utilize the hierarchical simulation method while taking into consideration of statistical performance distributions, we propose an efficient macromodeling method to handle this difficulty. The key aspect of the proposed macromodeling techniques is the extraction of parameterized behavioral models that can truthfully map the device-level variabilities to variabilities at the system level, so that the influence of fabrication stage variations can be propagated to the PLL system performances.

Parameterization can be done for each building block model as follows. First, multiple behavioral model extractions are conducted at multiple parameter corners, possibly following a particular design-of-experiments (DOE) plan [14]. Then, a parameterized behavioral model is constructed by performing nonlinear regression over the models extracted at different corners. This detailed parametric modeling step is advantageous since it systematically maps the device-level parametric variations to each of the behavioral models. However, difficulties arise when the number of parametric variations is large, which leads to prohibitively high parametric model extraction cost. We address this challenge by applying design-specific parameter dimension reduction techniques as described in the following section.

2. Hierarchical PLL Modeling

In this section we first describe the nominal behavioral model extraction for each PLL building block, then we discuss how a parameterized model can be constructed in the

next section.

a. Voltage Controlled Oscillator

The voltage controlled oscillator (VCO) is the core component of a PLL. The two mainstream types of VCOs are LC-tank oscillators and ring oscillators. A 5-stage ring oscillator and a VCO behavioral model are shown in Fig. 11. In this VCO model, the dynamic and static characteristics of the voltage to frequency transfer are modeled separately first and then combined to form the complete model.

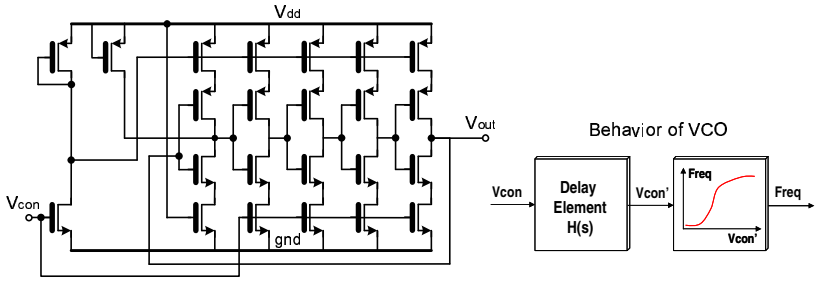

Fig. 11. Schematic of a ring oscillator and VCO macromodel.

As shown in the figure, the static VCO characteristic can be written as $F_{out} = f(V'_{con})$, where F_{out} is the output frequency, V'_{con} is the delayed control voltage, and $f(\cdot)$ is a nonlinear mapping relating the voltage with the frequency. $f(\cdot)$ can be further represented by an n-th order polynomial function

$$F_{out} = a_0 + a_1 \cdot V'_{con} + a_2 \cdot {V'_{con}}^2 + \cdots + a_n \cdot {V'_{con}}^n, \tag{2.6}$$

where $a_0, a_1, \cdots, a_n$ are the coefficients of the polynomial. To generate the above polynomial, multiple VCO steady-state simulations are conducted at different control voltage levels and a nonlinear regression is performed using the collected simulation data. The dynamic behavior of the VCO is modeled by adding a delay element that produces a delayed version of the control voltage, V'_{con}. The delay element can be

expressed using a linear transfer function $H(s)$ (e.g. a second-order RC network consisting of two R's and two C's).

b. Charge Pump

The schematic of a widely-used charge pump is shown in Fig. 12. The control signals of the two switches come from the outputs of the phase and frequency detector. The currents through M_1 and M_2 can be turned on-and-off to provide desired charge-up or charge-down currents.

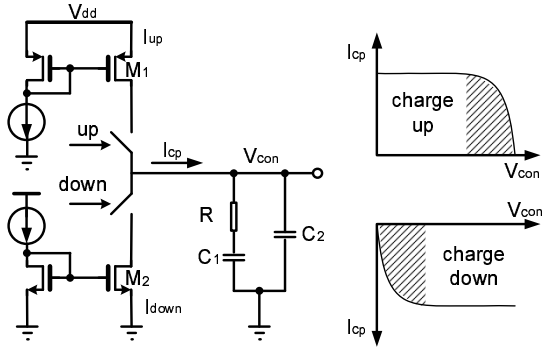

Fig. 12. Schematic of charge-pump.

The existing charge pump macromodels are very simplistic. Usually, both the charge-up and charge-down currents are modeled as constant values [16, 17]. A constant mismatch between the two currents may be also considered [15]. However, this simple approach is not sufficient to model the behavior of charge pump accurately. As indicated in Fig. 12, the current sources are implemented using transistors so that the actual output currents will vary according to the voltages across these MOSFET (M_1 and M_2). Therefore, the dependency of charge-up and charge-down currents on V_{con} must be considered.

In our charge pump model, for each output current, the current vs. V_{con} characteristics is divided into two regions. When the output voltage V_{con} is close to the

supply voltage, then M_1 will be biased in triode region. The charge-up current I_{up} in the triode region (shadow in Fig. 12) is given as

$$\begin{aligned} I_{up} &= \mu_p C_{ox} \frac{W}{L}[(V_{gs} - V_{thp})V_{ds} - 0.5V_{ds}^2] \\ V_{ds} &= V_{dd} - V_{on} - V_{con} \end{aligned} \tag{2.7}$$

where V_{dd} is the supply voltage, V_{on} is the on-voltage across the switch, V_{gs} is the gate-source voltage, μ_p is the mobility, C_{ox} is the oxide capacitance, W is width and L is the length of M_1. We can see from Eqn. 2.7 that the charge-up current is dependent on the output voltage V_{con}. We use a polynomial to explicitly model such voltage dependency

$$I_{up} = b_0 + b_1 \cdot V_{con} + b_2 \cdot V_{con}^2 + b_3 \cdot V_{con}^3 \tag{2.8}$$

where $b_i's$ are the polynomial coefficients. Similarly, the charge-down current has a strong V_{con} dependency when V_{con} is low. This voltage dependency is modeled in a similar fashion. When M_1 and M_2 operate in saturation region, they act as part of the current mirrors. In this case, constant output current values are assumed while the possible mismatch between the two are considered in our Verilog-A models.

c. Other PLL Circuit Blocks

The phase detector and the frequency divider are digital circuits so that they are more amenable to behavioral modeling. The two key parameters of the phase detector and the frequency divider are the output signal delay and the transition time, which are easy to extract from transistor-level simulation. The loop filters are usually comprised of passive RC elements, which can be directly modeled using Verilog-A. The complete PLL macromodel generation flow is shown in Fig. 13.

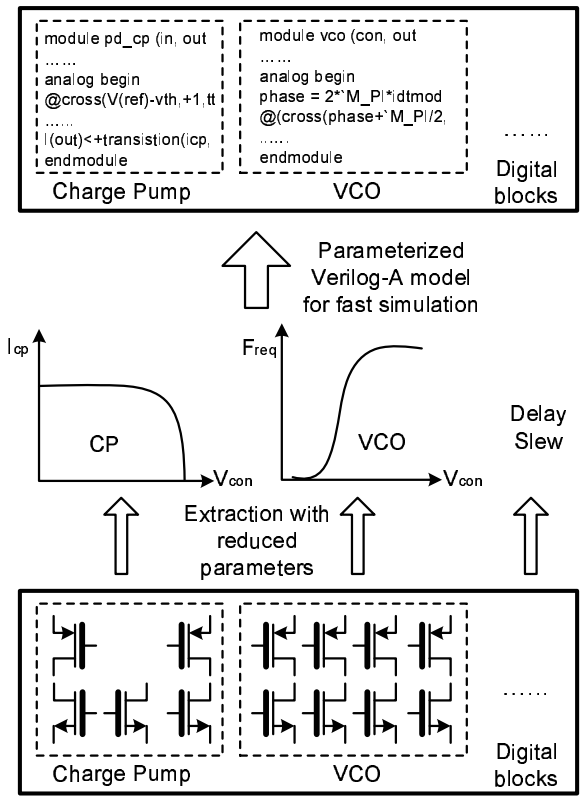

Fig. 13. PLL macromodel generation flow.

3. Efficient parametric-reduction PLL Modeling

When process variations are considered, parameterization can be done for each building block model as follows. First, multiple behavioral model extractions are conducted at multiple parameter corners, possibly following a particular design-of-experiments plan. Then, a parameterized behavioral model is constructed by performing nonlinear regression over the models extracted at different corners. This detailed parametric modeling step is advantageous since it systematically maps the device-level parametric variations to each of the behavioral models. However, difficulties arise when the number of parametric variations is large, which leads to a prohibitively high parametric model extraction cost.

The key parametric variations for a single transistor may include variations in mobility μ, gate oxide T_{ox}, threshold voltage V_{th}, effective length L_{eff} and so on [18]. The consideration of all possible sources of variations in transistors and interconnects can easily lead to explosion of the parameter space, rendering the parametric modeling infeasible. Although the widely used principle component analysis (PCA) [19] can be adopted to perform parameter dimension reduction, its effectiveness may be rather limited since parameter reduction is achieved by only considering the statistics of the controlling parameters while neglecting the important correspondence between these parameters and the circuit performances of interest. As such, the extent to which the parameter reduction can be achieved is not sufficient for our analog macromodeling problems. To address this difficulty, a more powerful design-specific dimension reduction technique, which is based on reduced rank regression (RRR), is developed. This new technique considers the crucial structural information imposed by the design and has been shown to be quite effective for parametric interconnecting modeling problems [20, 21].

a. RRR Based Parameter Dimension Reduction

Suppose we have a set of n process variations, X, and a set of N performances, Y. The objective is to identify a smaller set of new variables Z, based on X, which are statistically significant to the performances of interest, Y. Without loss of generality, let us assume Y nonlinearly depends on X through a quadratic model

$$Y = f(X) \approx [\beta_1 \;\; \beta_2] \begin{bmatrix} X \\ X \otimes X \end{bmatrix} \tag{2.9}$$

where β_1 and β_2 are the first and second order coefficients, $X \otimes X$ represents the quadratic terms of X. The combination of the linear and quadratic terms of X are then defined as a new predictor vector $\hat{X} = [X^T (X \otimes X)^T]^T$. Now the quadratic model in (2.9) can be cast into a linear model in $\hat{X}$ as: $Y = A\hat{X} + \varepsilon$. To identify the redundancy in X to facilitate parameter reduction, we seek a reduced rank regression model in the form

$$Y = A_R B_R \hat{X} + \varepsilon, \tag{2.10}$$

where A_R and B_R have a rank of R ($R < n$), and B_R has only R columns. We denote the covariance matrix of $\hat{X}$ as $Cov(X) = \Sigma_{\hat{X}\hat{X}}$, and covariance matrix between $\hat{X}$ and Y as $Cov(Y, \hat{X}) = \Sigma_{Y\hat{X}}$. It can be shown that an optimal reduced rank model (in the sense of mean square error) is given as [21]

$$A_R = U, B_R = U^T \Sigma_{Y\hat{X}} \Sigma_{\hat{X}\hat{X}}^{-1}, \tag{2.11}$$

where U contains $R(R < n)$ normalized eigenvectors corresponding to the R largest eigenvalues of the matrix: $D = \Sigma_{Y\hat{X}} \Sigma_{\hat{X}\hat{X}}^{-1} \Sigma_{\hat{X}Y}$. It is important to note that a successful construction of the above reduced rank model indicates that only a smaller set of R new parameters $Z = B_R \hat{X}$ are critical to Y in a statistical sense, hence facilitating

the desired parameter reduction.

It should be noted that the reduced rank regression is only employed as a means for parameter reduction so as to reduce the complexity of the subsequent parameterized macromodeling step. Hence, Y in the above equations do not have to be the true performances of interest and can be just some circuit responses that are highly correlated to the performances. This flexibility can be exploited to more efficiently collect $\Sigma_{Y\hat{X}}$ though Monte-Carlo sampling if Y are easier to obtain than the true performances in simulation.

b. Parameterized Macromodeling Using Parameter Reduction

The parameterized PLL macromodel extraction flow is shown in Fig. 13. As mentioned in Eqn. 2.6 and 2.8, each behavioral model parameter is expressed as a polynomial in the underlying device-level variations, such as

$$\beta = f(V_{th1}, L_{eff1}, T_{ox1}, \cdots, V_{thn}, L_{effn}, T_{oxn}), \tag{2.12}$$

where V_{thi}, L_{effi}, T_{oxi}, etc represent the parameters of i-th transistor, $f(\cdot)$ is the nonlinear polynomial function. $f(\cdot)$ is very difficult to obtain if the number of parameters is large. Hence, RRR-based parameter reduction is applied, which leads to a set of R new parameters Z that are the most important variations for the given circuit performances of interest. If R is small, then a new parameterized model in terms of Z can be easily obtained through conventional nonlinear regression

$$\beta = \hat{f}(Z_1, Z_2, \cdots, Z_R). \tag{2.13}$$

In addition to reducing the cost of parameterized macromodeling, parameter dimension reduction also leads to more efficient statistical simulation of the complete PLL. This is because instead of analyzing the design performance variations over

the original high-dimensional parameter space, statistical simulation can be now performed more efficiently in a much lower dimensional space that carries the essential information of the design variability. The application of the parameter reduction can also improve the efficiency of DFT schemes as to be described in Chapter IV.

C. General Block Modeling Using Kriging Models

We have presented the efficient modeling techniques for Sigma-Delta ADCs and charge-pump Phase-locked Loops. The ability to accurately model arbitral analog circuits are important and necessary for automatic circuit performance optimization. The complex nature of analog/mixed-signal systems, however, makes this task difficult and costly. In this section, we adopt a *Geostatistics motivated* approach (i.e. Kriging model) for efficient generation of performance models considering both design and process variables for analog circuits [6]. Kriging model is attractive for our yield-aware analog performance modeling problem because of two appealing features. First, Kriging model enables robust regression of global trends of complex mapping between design parameters and resulting performances. The capability in capturing global trends of the performance space is very beneficial as it helps prevent trapping into local optimum that may happen in optimization-based approaches. Second, in addition to providing performance predictions, Kriging model also provides an *uncertainty level* for each prediction in the form of mean-square-error (MSE). The ability in providing such an assessment on prediction accuracy immediately allows an iterative update scheme wherein new data can be selectively added in the regions of high uncertainty level to improve the model accuracy. We further utilize the obtained performance models of building blocks to achieve efficient hierarchical system optimization in the next chapter.

1. Mathematical Formulation

Kriging model was first proposed by Matheron in 1963 [6] for geostatisics problems. Unlike physical experiments, computer simulation does not posses any random errors. However, under the framework of Kriging, a deterministic circuit performance using Kriging model is considered as a realization stochastic process, $Y(x)$, where x is an n-dimensional vector containing design parameters (and/or process variations). This fundamental treatment of Kriging model provides a statistical framework for deterministic function approximation and quantification of approximation uncertainty [22]. The stochastic process $Y(x)$ is cast into a regression model as

$$Y(x) = Z(x) + \beta^T \mathbf{f}(x) \tag{2.14}$$

where $\mathbf{f}(x) = [f_1(x)\ f_2(x)\ \cdots\ f_n(x)]^T$ is a vector of predefined regression functions, $\beta = [\beta_1\ \beta_2\ \cdots\ \beta_n]^T$ is the vector of unknown regression coefficients. The term $\beta^T \mathbf{f}(x)$ represents the global trend of $Y(x)$ across the input space and $Z(x)$ is a random process and used to capture the systematic departure of the performance from the global regression portion. $Z(x)$ is assumed to have zero mean and a correlation matrix

$$Corr(Z(x_i), Z(x_j)) = R(x_i, x_j) \tag{2.15}$$

for $Z(x_i)$ and $Z(x_j)$ with two input vectors x_i and x_j. The correlation matrix in Eqn 2.15 is often parameterized and can be chosen to be a product of stationary one-dimensional functions

$$R(x_i, x_j) = \prod_{k=1}^{n} e^{-\theta_k |x_{i,k} - x_{j,k}|^{p_k}} \tag{2.16}$$

where $\theta = [\theta_1\ \theta_2\ \cdots\ \theta_n]^T$ and $p = [p_1\ p_2\ \cdots\ p_n]^T$ are unknown coefficients with constrains $\theta_k \geq 0$ and $0 \leq p_k \leq 2$.

In order to apply Eqn 2.14 for performance modeling, we need to estimate the unknown parameters σ^2, β, p and θ. Suppose m sets of simulations are performed, we have $X_s = [x_{s,1},\ x_{s,2},\ \cdots\ x_{s,m}]^T$ as the input vectors (can be design variables or process variables) and $\mathtt{Y}_s = [\mathtt{y}_{s,1},\ \mathtt{y}_{s,2},\ \cdots\ \mathtt{y}_{s,m}]^T$ be the corresponding performances. An $m \times m$ matrix R is defined by $R_{i,j} = R(x_i, x_j)$. The goal of Kriging model is to find a predictor of the system performances at the new point x_{new}.

If we denote $r(x_{new}) = [R(x_{new} - x_1)\ R(x_{new} - x_2)\ \cdots\ R(x_{new} - x_m)]^T$ and $F(x) = [f(x_1)\ f(x_2)\ \cdots\ f(x_m)]^T$, the predictor $\hat{Y}(x_{new})$ can be written as [23]

$$\hat{Y}(x_{new}) = f(x_{new})^T\hat{\beta} + r^T R^{-1}\left(Y_s - F\hat{\beta}\right) \tag{2.17}$$

where

$$\hat{\beta} = \left[F^T R^{-1} F\right]^{-1} F^T R^{-1} Y_s. \tag{2.18}$$

The MSE of the predictor $\hat{Y}(x_{new})$ and the real performance value can be obtained as

$$MSE[\hat{Y}_0] = \hat{\sigma}^2\left(1 - \begin{bmatrix} f^T & r \end{bmatrix}\begin{bmatrix} 0 & F^T \\ F & R \end{bmatrix}^{-1}\begin{bmatrix} f(x_{new}) \\ r(x_{new}) \end{bmatrix}\right) \tag{2.19}$$

where $\hat{\sigma}^2$ is given as

$$\hat{\sigma}^2 = \frac{1}{m}\left[Y_s - F\hat{\beta}\right]^T R^{-1}\left[Y_s - F\hat{\beta}\right]. \tag{2.20}$$

To get the performance prediction, the Maximum likelihood estimate (MLE) of correlation parameters θ and p as well as β and σ^2 is performed. The likelihood function can be written as

$$\begin{aligned} L(\theta, p, \beta, \sigma^2) = {} & \tfrac{1}{(2\pi)^{m/2}} \cdot \tfrac{1}{(\sigma^2)^{m/2}} \cdot \tfrac{1}{(\det(R))^{1/2}} \\ & \cdot \exp\left(-\tfrac{1}{2\sigma^2}\left[Y_s - F\beta\right]^T R^{-1}\left[Y_s - F\beta\right]\right) \end{aligned}. \tag{2.21}$$

Substituting Eqns 2.18 and 2.20 into Eqn 2.21 leads to the log of the likelihood

function

$$L(\theta, p) = -\frac{1}{2}\left[m \log \hat{\sigma}^2 + \log(\det(R))\right]. \tag{2.22}$$

In practical implementation, we can first select a proper value for p (e.g. 2), and solve Eqn 2.22 numerically to find $\hat{\theta}$. Then, Eqn 2.18 and Eqn 2.20 are used to compute $\hat{\beta}$ and $\hat{\sigma}^2$. Finally, the estimated performance at x_{new} is given by Eqn 2.17 and the MSE of $\hat{Y}(x_{new})$ is given by Eqn 2.19, which can be used to evaluate the prediction accuracy. A relative prediction uncertainty is more meaningful

$$Err = \left(MSE[\hat{Y}_0]\right)^{1/2} / \hat{Y}_0. \tag{2.23}$$

We summarize the application of Kriging model for circuit performance modeling in Algorithm. 1. Here the goal is to predict the system performance P at any design point D in the design space.

Algorithm 1 Generation of Kriging Performance Model

Input: Circuit netlist, target performance P, design variables $D = [d_1, \cdots, d_n]$, max relative error tolerance Err_{max}, # of test points n.
Output: The Kriging model $P = K(D)$.

1: Evenly sample the design space using m date points $D_1, \cdots, D_m$ and find the corresponding performances $P_1, \cdots, P_m$ via SPICE simulation.
2: Construct an initial Kriging model $P = K(D)$ by maximizing Eqn 2.22.
3: **for** $i = 1$ to n **do**
4: Randomly generate a design sample $D_{T,i}$ and evaluate the relative MSE Err_i using Eqn 2.23 for $\hat{P}(D_{T,i})$;
5: **if** $Err_i > Err_{max}$ **then**
6: Perform a SPICE simulation at $D_{T,i}$ to compute the exact performance $P_{D_{T,i}}$.
7: Include $\{D_{T,i}, P_{D_{T,i}}\}$ as an additional data and update the Kriging model.
8: **end if**
9: **end for**

The above algorithm iteratively constructs a Kriging model for a single performance. The model is updated in the areas where the Kriging models do not possess of enough accuracy, which leads to the iterative model updating feature. If more

than one performance are of interest, multiple Kriging models can be extracted individually. With the help of Kriging performance models we can achieve yield-aware hierarchical analog/mixed-signal system optimization and also help develop efficient built-in self-test circuits, which will be discussed in detail in the next chapters.

2. Circuit Examples

In order to demonstrate the accuracy and efficiency of the proposed Kriging performance methodology, we use two test circuit cases including one ring oscillator and one LC oscillator which are implemented in 90-nm CMOS technology.

a. Ring Oscillator

The first circuit under analysis is a typical five-stage ring oscillator similar as in Fig. 11. The total number of transistors in this circuit is 23. We consider the sizes (width W with length set to the minimum) of all the transistors with symmetry constrains, that is the inverters should be identical across different stages. Therefore, there are 7 design parameters in our design example. Threshold voltage V_{th} of all the 23 transistors are assumed to follow Gaussian distributions to model the process variations. The circuit performance considered are power, maximum frequency and VCO gain at the center operation frequency.

We set the following design constrains: each transistor width can be varied within $\pm 40\%$ of the initial design value and should be at least greater than the minimal transistor width. The variation of V_{th} is set to $3\sigma = 15\%$ to account for the process variations. The initial Kriging model is built with 50 uniform samples in the design space. After the generation of the Kriging model, we compare the predicted performance and corresponding predicted relative MSE with the SPICE simulation results for four randomly selected points as shown in Table IV. Y_p is the predicted performance, Y_a

is the measured results by SPICE, MSE is the predicted relative error and Err is the actual relative error of Y_p w.r.t. Y_a. From Table IV, we can clearly see that the

Table IV. Kriging model accuracy for ring oscillator.

$Freq_{max}$	$Y_p(GHz)$	$Y_a(GHz)$	$MSE(\%)$	$Err(\%)$
Point 1	2.768	2.782	0.53	-0.53
Point 2	2.462	2.440	0.36	0.90
Point 3	2.164	2.151	0.41	0.60
Point 4	1.878	1.861	0.66	0.92
Power	$Y_p(\mu W)$	$Y_a(\mu W)$	$MSE(\%)$	$Err(\%)$
Point 1	51.98	51.91	0.29	0.14
Point 2	42.50	42.72	0.41	-0.50
Point 3	39.85	39.89	0.23	-0.099
Point 4	36.54	36.79	0.48	-0.69
Gain	Y_p (G/V)	Y_a (G/V)	$MSE(\%)$	$Err(\%)$
Point 1	5.607	5.671	0.54	-1.14
Point 2	4.823	4.796	0.47	0.56
Point 3	4.489	4.517	0.70	-0.63
Point 4	4.100	4.105	1.31	-0.11

predicted performances match well with the actual measured performance values.

b. LC Oscillator

The second circuit example is an LC oscillator as shown in Fig. 14. There are totally six transistors including two acting as a variator. We use R_1 and R_2 to model the parasitics resistances of the two inductors L_1 and L_2, and C_{f1} and C_{f2} to model the

parasitic capacitances at the two output nodes.

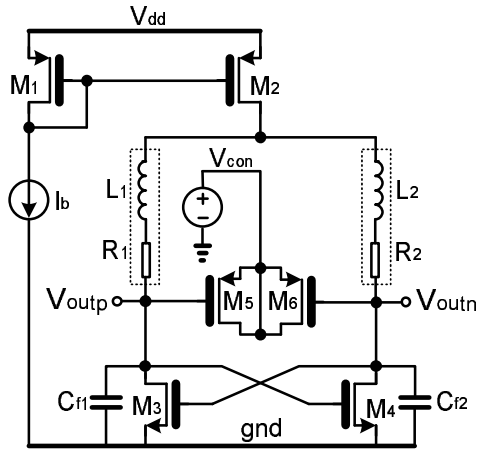

Fig. 14. LC oscillator schematic.

Width W of all the transistors are considered as design parameters with symmetry constrains between M_3 and M_4, M_5 and M_6, respectively. The biasing current I_b is also selected as a design parameter. In terms of process variation modeling, we consider the threshold voltage V_{th} of each transistor and the mismatch between inductors L_1 and L_2, the parasitic capacitances and resistances are sources of variability. The ranges of the transistor widths and the bias current are set to be within $\pm 30\%$ of the initial design, and transistor widths should also satisfy the minimal width requirement. The variations of V_{th} and parasitic parameters are set to be $3\sigma = 10\%$.

We consider the VCO gain and the power for this LC oscillator and build Kriging models to predict the system performance. The accuracy of the Kriging model $\overrightarrow{K}(\overrightarrow{D}, \overrightarrow{V})$ is demonstrated in Table V, in which we pick up four random combinations of the design parameters $\overrightarrow{D}$ and process variations $\overrightarrow{V}$.

Table V. Kriging model accuracy for LC oscillator.

Power	$Y_p(\mu W)$	$Y_a(\mu W)$	$MSE(\%)$	$Err(\%)$
Point 1	55.25	55.17	0.76	0.14
Point 2	50.42	50.44	0.07	-0.03
Point 3	56.90	56.78	0.11	0.21
Point 4	50.18	50.15	0.13	0.07
Gain	Y_p (G/V)	Y_a (G/V)	$MSE(\%)$	$Err(\%)$
Point 1	1.107	1.106	0.83	0.09
Point 3	1.250	1.249	0.54	0.08
Point 2	1.316	1.294	0.77	1.67
Point 4	1.071	1.071	1.24	0.02

D. Summary

In this chapter, we propose an efficient lookup table based modeling technique for Sigma-Delta ADC performance evaluations. By combining response surface modeling technique, parameterized LUT models in terms of important underlying circuit parameters are constructed. Our modeling framework can be employed to perform fast nominal and statistical simulation of various Sigma-Delta ADC designs and provide a basis for performance and robustness trade-off analysis. We also look into the modeling of charge-pump PLLs by presenting a macromodeling strategy with reduced-rank regression to perform fast statistical PLL performance evaluations. The PLL modeling frameworks are used to efficiently evaluate the PLL performance distributions under process variation. A geostatistics motivated yield-aware modeling approach is presented for general analog circuit performance modeling. The MSE metric provided by the Kriging model is instrumental in controlling modeling accuracy via

well-controlled model updates. Experimental results are presented to demonstrate the effectiveness of the proposed performance modeling approaches. The proposed performance modeling approaches are critical for the analog/mixed-signal system optimizations and performance verifications, as to be discussed in the following chapters.

CHAPTER III

YIELD-AWARE ANALOG CIRCUIT OPTIMIZATION

Optimization of large analog/mixed-signal systems is difficult due to the costly system performance evaluation procedures and the large design variable space. The optimization task becomes even more challenging when process variations come into consideration. In this chapter we utilize Kriging performance models to first achieve circuit block level performance trade-offs and then use these models to facilitate yield-aware hierarchical system optimization [24].

Design centering technique is a widely used yield enhancement approach by moving the nominal designs to increase the overlap of the process distribution and the feasible performance space [25]. Although the idea is simple and straightforward, the implementations of design centering techniques could vary. One variant of the design centering implementation uses linear model with sensitivity analysis to form an approximation of the overall feasibility region as polytope [26]. Other implementations may treat the performance feasible region as an ellipsoid rather than a convex polytope, then the optimization target becomes to move the final design to the center of the ellipsoid [27]. The major drawback of these simplifications is the accuracy, simple linear models have very poor capability to capture the complex process variation behaviors in modern CMOS technologies, so improved techniques are needed for paretical yield-aware analog optimizations.

One of the two most popular commercial yield-enhance tools for analog circuit designs is NeoCircuit by Cadence Design Systems [28]. It employs Monte-Carlo simulations to find the process parameter combinations which corresponds to the worst system performances, then the tool performs design space optimization to meet these process corners. The underlining theory is simple, if the design can meet the spec-

ifications in the worst performance corners, it can meet the specifications in all the other process variation conditions, hence the design is optimized in the center. The other analog optimization tool, WiCkeD developed by MunEDA, utilizes the worst case distance to guide the optimization search [29]. In that optimization framework, the major difference from NeoCircuit is that the process corners used to guide the yield optimization are not from Monte-Carlo simulations but from the numerical calculation of the worst-case distance [30].

There exist several drawbacks in these traditional design centering techniques for analog yield optimization. First of all, no matter model-based or corner-based yield analysis techniques, they can only maintain the accuracy for the specified local design points. Once the design points are shifted during the system optimization, process models or corners need to be updated in the new design points, hence no optimization convergence can be guaranteed. Second, the yield analysis are rough in these optimization frameworks. The accuracy of the yields calculated using process space models heavily depend on the model accuracy and the shape of process distributions, while the corner based approaches can not specify the required yield levels. And lastly, these traditional optimization methods lack of the capabilities to address the problems of large scale analog circuit optimization.

In order to address the limitations of these traditional analog optimization approaches, we propose to use the accurate and efficient global modeling technique, the Kriging modeling method, as the key element in our yield-aware analog optimization framework. The Kriging model bridges the design and process variable spaces and the system performance spaces. The performance models are updated iteratively in the whole design space to enhance the accuracy, so we can use the Kriging performance models to search for the global optimal solutions. The statistical system performances are analyzed using the device-level Monte-Carlo simulations which are evaluated us-

ing the efficient partial Kriging models, so the yields calculated are accurate. The proposed optimization framework keeps the structure of efficient hierarchical system performance analysis so the computation time of the performance evaluation within each optimization iteration can be reduced significantly. In this chapter we start with the optimization of smaller scale analog circuits with the consideration of process variations, then present the more challenging hierarchical yield-aware analog system optimization framework.

A. Yield-aware Circuit Block Optimization

The mathematical forms of Kriging models have been presented in the previous Chapter. Suppose we denote the Kriging performance model as $\overrightarrow{K}(\overrightarrow{D}, \overrightarrow{V})$ where $\overrightarrow{D}$ is the design parameter set and $\overrightarrow{V}$ represents the process variation information. Performances of small scale analog/mixed-signal circuits can be modeled accurately using Kriging model, so we can perform circuit optimization using $\overrightarrow{K}(\overrightarrow{D}, \overrightarrow{V})$ by searching in the design space $\overrightarrow{D}$ while injecting process uncertainties using $\overrightarrow{V}$.

In this section, we first evaluate circuit performance trade-offs (pareto fronts) without considering process variations using iterative search with nominal Kriging performance models. The generated nominal building block pareto fronts serve as the baseline to generate the yield-aware performance trade-offs. Then we use the nominal pareto fronts as starting points to search for the performance trade-offs in different yield levels achieving better convergence. The yield-aware optimization is obtained by optimizing system cost functions consisted of all the system performances in the specified yield levels. The statistical system performance distribution analysis is achieved using partial Kriging models, which save about 60% to 70% computation time when compared with brute-force use of Monte-Carlo simulation. We demonstrate

the efficiency and accuracy of the proposed yield-aware optimization framework using design examples of two oscillator and one amplifier.

1. Pareto Front Background

In most circuits, different performance objectives compete against each other and it is infeasible to find a design point to reach the best value for all performances at the same time. The design task then becomes a multi-objective optimization problem. As shown in Fig. 15, pareto front consists of the optimal performance trade-offs between different performances. In multi-objective optimization, performance p_a dominates performance p_b (suppose smaller value is better) when [31, 32]

$$\mathbf{p_a} \prec \mathbf{p_b} : \forall(p_{a_i} \leq p_{b_i}) \wedge \exists(p_{a_i} < p_{b_i}), \quad i = 1, \cdots, n \tag{3.1}$$

where p_{a_i} and p_{b_i} are the i-th performances of interest, and there are totally n performances. A set of performances is considered as pareto-optimal if it is not dominated by any other set of performances.

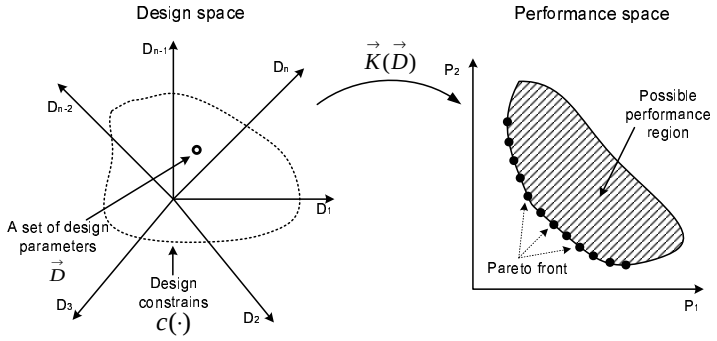

Fig. 15. Illustration of pareto front.

In practice, it is impossible to obtain the entire pareto front in the closed form and it has to be approximated by finding a number of points on the front. So the multi-objective optimization problem is often transformed to multiple single-objective

optimization problems by assigning a suitable weight to each performance [31, 32].

2. Iterative Search for Pareto Fronts

With the definition in Eqn. 3.1, we start to build pareto fronts without considering process variations by employing nominal Kriging performance models. Although Kriging model provides a general performance modeling machinery, its practical application in circuit modeling must be facilitated by well controlling accuracy and complexity. The key idea is to develop an iterative model accuracy refining scheme so that the pareto fronts generated represent the actual circuit performance trade-offs. Our strategy help control the total number of transistor-level circuit performance evaluations (SPICE simulations) by exploiting the prediction power of Kriging model and adopting incremental updates.

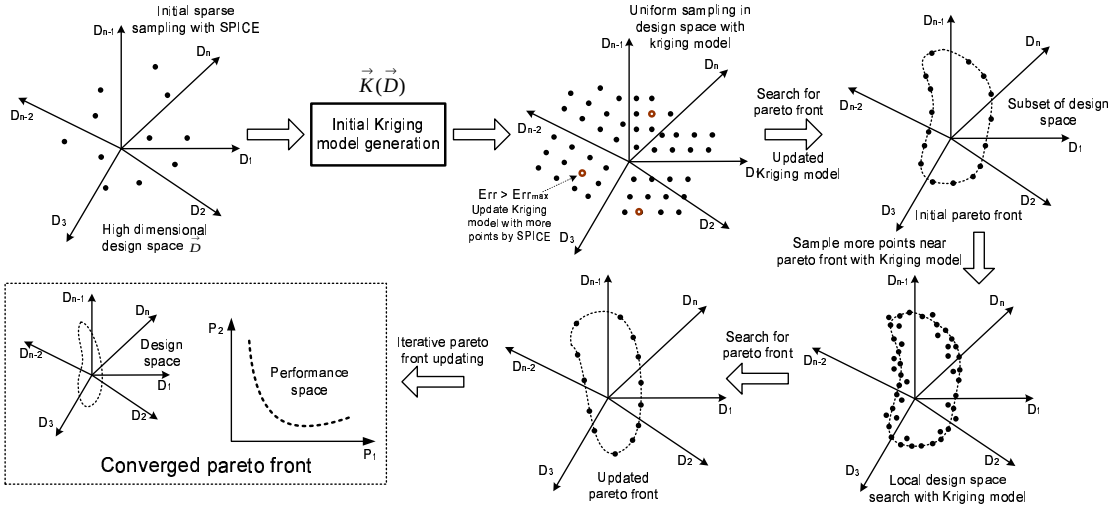

Fig. 16. Iterative pareto front generation.

1) As shown in Fig. 16, our nominal-case iterative pareto front extraction starts from a coarse sampling of design space $\overrightarrow{D}$ by using a limited number of SPICE simulations. This initial set of SPICE simulation data serves as a basis of constructing an initial Kriging model as in Eqn 2.17 for predicting circuit performances.

2) In the following step, the complete design space is sampled uniformly to capture the global trend of design parameter to performance mapping through multiple evaluations of the Kriging model. The Kriging model accuracy is improved by adding additional SPICE simulation data when the relative MSE in Eqn 2.19 exceeds a user-defined threshold. In our experiments, it has been observed the cost of these selective SPICE simulations is rather mild.

3) With a conditionally updated Kriging model and the set of Kriging model evaluations in the previous step, an initial pareto front is extracted by sorting the evaluated performance values. The cost due to sorting is usually insignificant.

4) As the initial pareto front only captures the global trends of design parameters to circuit performances mappings, a local design space search is conducted around the initial pareto front to refine the initial pareto front. A number of additional design points in the neighborhood of the initial pareto front are evaluated using the Kriging model. Note that this local design space exploration is not expensive since performance evaluation is achieved through Kriging.

5) The pareto front is updated by examining performances of all the design points involved in the local design space search, and this iterative process continues till the a converged pareto front is reached. It has been observed that the convergence can be usually reached in a few iterations in our experiments.

The obtained nominal pareto fronts serve as the baseline for the yield-aware pareto front generation. Although the "optimal" performance tradeoffs associated with the nominal case pareto front often represent overly optimistic performance combinations under significant process variations, one *key* observation is that the design parameters on the nominal pareto front are often still the near-optimal design candidates in the presence of process variations. This allows us to use the nominal pareto front as the starting point and adopt a well-controlled iterative process to

search for pareto fronts under varying yield levels.

3. Fast Statistical Analysis Using Partial Kriging

Although Kriging model is a much more efficient surrogate than circuit simulation, a brute-force use of Kriging model to run Monte-Carlo simulations could be rather expensive. Our key idea to alleviate the cost is to facilitate computation sharing between Monte-Carlo simulations across the design space via *Partial Kriging Model Evaluation*. This is achieved by properly choosing a correlation function (i.e. Eqn 2.16) and exploiting special structure of the Kriging model.

Let us examine Eqn 2.17 that provides the circuit performance predication by the Kriging model. Without loss of generality, in the following discussion, we only concern with a scalar performance. In the practical implementation, $f(\cdot)$ can be simply chosen as constants, then only variable in Eqn 2.17 is vector r, which corresponds to the correlations between the untried input x_0 and the m inputs $x_{s,1}, \cdots, x_{s,m}$ based on which the Kriging model is built. Eqn 2.17 is now written as

$$\hat{Y_0} = c + r^T q, \tag{3.2}$$

where c is a constant and $q = R^{-1}(Y_s - F\hat{\beta}) \in \mathbf{R}^m$, both of which are known after the Kriging model is constructed. The main cost of a Kriging model evaluation is due to the computation of r that involves of computation of nm exponential terms and their products as in Eqn 2.16, where m is the range of a few hundreds and n is the dimension of the combined design and process parameter space, which is typically in the range of a few tens. The computation of r takes the most significant portion of model evaluation time.

Our key observation is that due to the specific structure of our Kriging model, the design and process variables are well separated in the evaluation of r. To see this

more clearly, we rewrite Eqn 2.16 as

$$R(x_0, x_i) = \prod_{k=1}^{N_D} e^{-\theta_k |x_{0,k} - x_{i,k}|^{p_k}} \cdot \prod_{k=N_D+1}^{N_D+N_V} e^{-\theta_k |x_{0,k} - x_{i,k}|^{p_k}}, \tag{3.3}$$

where N_D is the number of the design parameters and N_V is the number of process parameters, and $N_D + N_V = n$. Note that the second product $R_r(x_0, x_i)$ of the above equation does not depend on the design parameters, so it can be pre-computed and shared for different design points. Essentially, before the yield-aware design space starts, we first perform one run of Monte-Carlo sampling in the process space by pre-computing a set of $R_r(x_0, x_i)$ samples. This set of samples do not complete the Kriging model evaluations solely by themselves, therefore this step is referred to as *Partial Kriging Model Evaluation*. This set of $R_r(x_0, x_i)$ samples can be shared with other design points to facilitate fast Monte-Carlo simulations in the design space efficiently. For instance, as shown in Fig. 17, for any give design point, we only need to evaluate

$$R_l(x_0, x_i) = \prod_{k=1}^{N_D} e^{-\theta_k |x_{0,k} - x_{i,k}|^{p_k}} \tag{3.4}$$

and combine it with the $R_r(x_0, x_i)$ samples to quickly generate the statistical performance distribution at this design point. Our technique is especially attractive in practice since under most cases $N_V > N_D$.

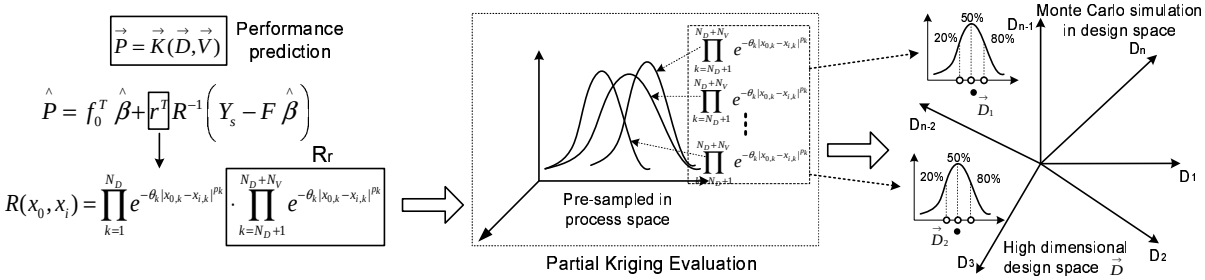

Fig. 17. Speeding up Monte-Carlo sampling via partial Kriging model evaluation.

4. Yield-aware Block Optimization

In the previous section, we have described how partial Kriging model evaluation can be exploited to facilitate efficient Monte-Carlo simulations in the design space based on Kriging models $\overrightarrow{K}(\overrightarrow{D}, \overrightarrow{V})$ that span across both the design and process spaces. However, the construction of such Kriging models can be considerably costly compared with the case of the nominal case modeling due to a higher input space dimensionality. We tackle this difficulty by including Kriging model construction as part of iterative pareto front extraction procedure where only localized updates and search are involved.

Although the "optimal" performance tradeoffs associated with the nominal case pareto front often represent overly optimistic performance combinations under significant process variations, one *key* observation is that the design parameters on the nominal pareto front are often still the near-optimal design candidates in the presence of process variations. This allows us to use the nominal pareto front as a starting point and adopt a well-controlled iterative process to search for pareto fronts under varying yield levels as shown in Fig. 18.

Starting from the nominal pareto front, we perform local design space search by evaluating additional design points in the neighborhood of the initial pareto front. Here, each design point is not only evaluated in the nominal sense, but also statistically by performing Monte-Carlo sampling in the process space. Hence, performances that can be achieved at varying yield levels (e.g. 60% or 80%) are obtained after the Monte-Carlo simulation. It shall be noted the cost of these Monte-Carlo simulation is well controlled by performing Partial Kriging Model Evaluation based fast technique. The relative MSE of each full Kriging model evaluation is also checked and the Kriging model is again conditionally updated by performing additional SPICE

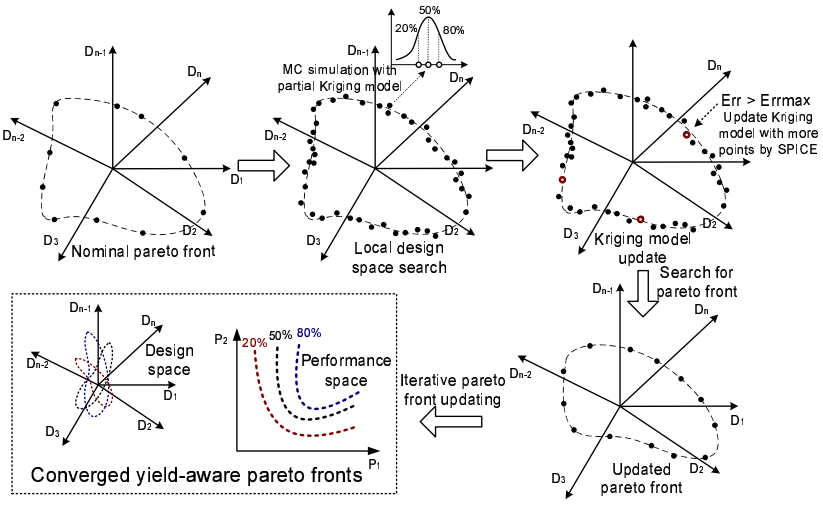

Fig. 18. Iterative yield-aware pareto front optimization.

simulation if the MSE exceeds the specified threshold. Next, for any required yield level α, performances that can be achieved at yield α in the local design space search are sorted and the pareto front is updated accordingly. This iterative process continues till a converged pareto front is reached for each yield level α. The achieved yield-aware pareto fronts represent the best achievable performance trade-offs in the required yield level.

5. Block Optimization Examples

In this section we first apply the yield-aware optimization method for the two design cases illustrated for Kriging model accuracy in Chapter II. The 50% yield and 80% yield pareto fronts as well as the nominal pareto fronts as in Fig. 19 for the ring oscillator in Fig. 11.

To verify that the pareto fronts we have obtained in Fig. 19 are globally optimal, we perform Monte-Carlo simulations in the neighborhoods of these found fronts and also in the far away regions in the design space, to check the optimality of our Pareto fronts. The pareto front with 50% yield is selected as an example and the simulation

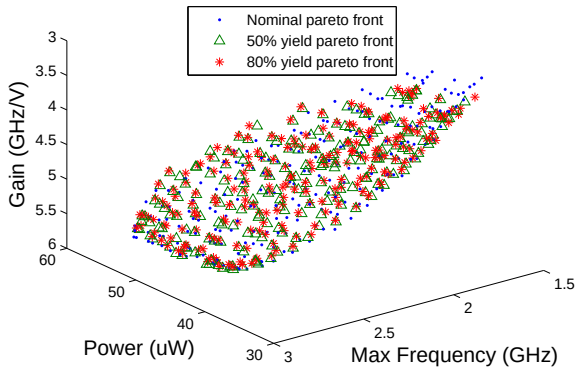

Fig. 19. Yield-aware pareto fronts for the ring oscillator.

results are shown in Fig. 20. In the figure, we examine the performance space region where the maximum frequency is at 2.4GHz $\pm$ 0.1GHz. As can be seen from the figure, the additional design points are inferior to the found pareto front, demonstrating the effectiveness of our yield-aware pareto front generation method.

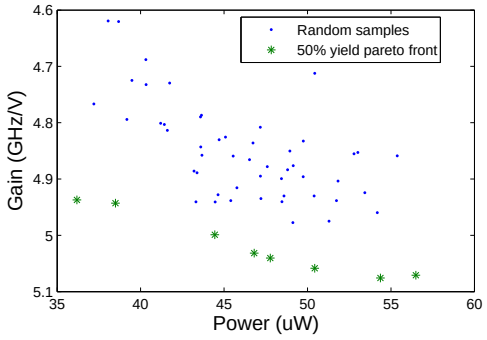

Fig. 20. Verification of yield-aware pareto front.

Similar as the ring oscillator example, we plot the pareto fronts for the LC oscillator in Fig. 21. Fig. 21 clearly shows the difference between the nominal pareto fronts and the pareto fronts at different yield levels. We can see that if a higher yield is needed, the corresponding performance trade-offs become worse.

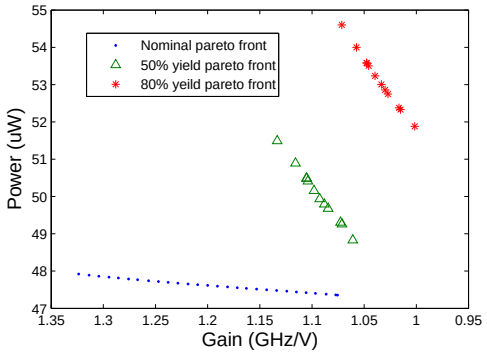

Fig. 21. Yield-aware pareto fronts for the LC oscillator.

We further perform optimization for a two-stage operational amplifier as shown in Fig. 22. There are totally 8 transistors in the circuit and we consider 5 design variables for the optimization and 8 transistor mismatch parameters to represent process variations, set up with the same range as those of the LC oscillator.

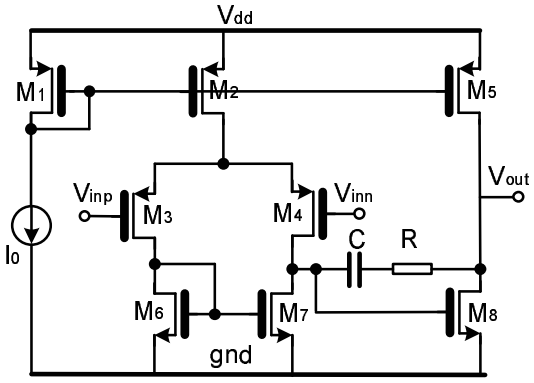

Fig. 22. Two-stage Op-Amp schematic.

The performance we select to generate pareto fronts are the DC gain and the 3-dB bandwidth. We show an intermediate step in our pareto front search algorithm and the converged front in Fig. 23.

The yield-aware pareto fronts of 20%, 50% and 80% are plotted in Fig. 24 for the two-stage opamp. The pareto front curve denote the best achievable circuit

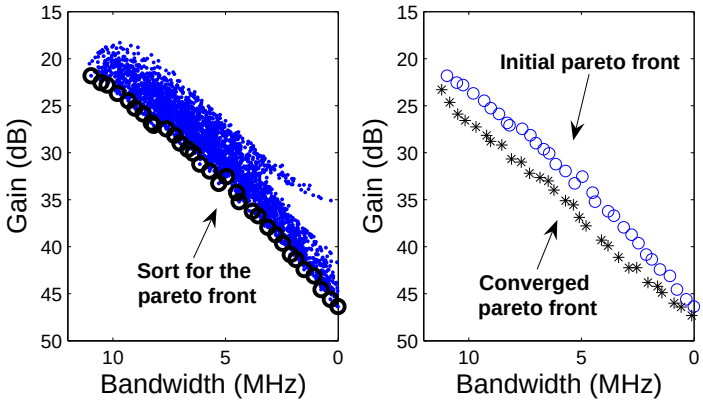

Fig. 23. Iterative pareto front generation for the two-stage Op-Amp.

performance trade-offs at the required yield levels.

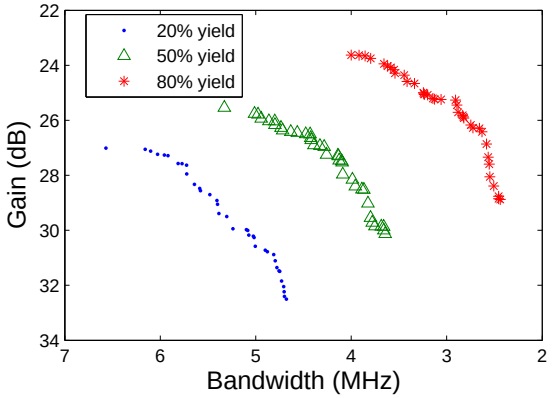

Fig. 24. Yield-aware pareto fronts for the two-stage Op-Amp.

B. Yield-aware Hierarchical System Optimization

The yield-aware hierarchical optimization is dictated by the need for safeguarding large analog/mixed-signal designs in scaled CMOS technologies. Although in the previous section we successfully achieved yield-aware synthesis for small scale analog circuits, it is of much more interest to solve the problem of automatic design for large analog systems. Hierarchical optimization is a promising approach for large

system designs. However, the existing flows are lack of the ability to consider process variations in the automatic system design, which is much needed to ensure the system yield.

We address two fundamental difficulties in achieving efficient robust analog system optimization: yield-aware pareto performance characterization at the building block level and yield-aware system-level optimization problem formulation. It is shown that the proposed approach is not only able to effectively capture the block performance trade-offs at different yield levels, but also correctly formulate the whole system yield and efficiently perform system-level optimization in presence of process variations. Our approach extends the efficiency of hierarchical analog optimization, enjoyed for improving nominal circuit performances, to yield-aware optimization. The proposed methodology is demonstrated by the two examples of a two-stage amplifier and a phased locked loop (PLL) consisting of multiple building blocks.

1. Hierarchical Optimization Background

In hierarchical optimization, a large analog system is decomposed into several building blocks. In order to get the best overall system performances, it is natural to find the design points which result in best performances for the building blocks. Pareto fronts presented in the previous section are the good representation of the performance trade-offs needed in hierarchical optimization framework. The optimal system performances can be achieved by searching within building block-level pareto fronts.

A general flowchart of hierarchical optimization using pareto fronts is shown in Fig. 25. Circuit level design parameters (sizes of transistor and passive components, biasing, etc) are explored within the design constrains to find the best possible performance trade-offs. Then system-level optimization is carried out by searching in the space restricted using block-level pareto fronts. If the mappings from system

performances to building block parameters and block-level variables to circuit-level parameters are known, we can retain the corresponding circuit design parameters to the optimal system performances. There exist two key benefits for this hierarchical optimization. First, since the number of performances in the block level is much smaller than that of the original design space, the space exploration can be reduced significantly. An equally important benefit here is that system-level behavioral models can be used to quickly estimate system-level performances, thereby further significantly reducing the overall optimization cost.

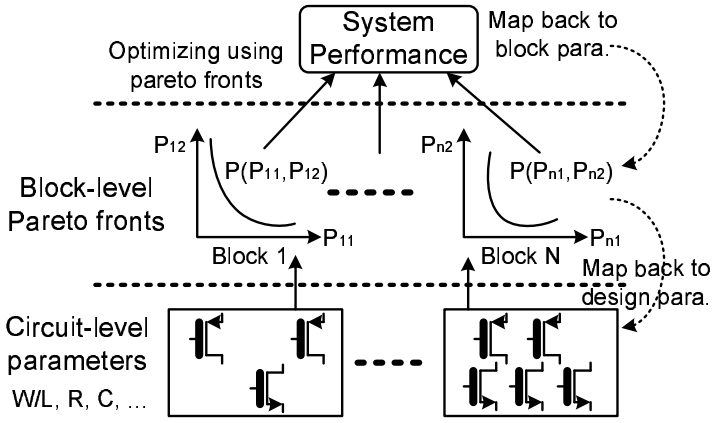

Fig. 25. Nominal hierarchical optimization flow.

2. Issues in Yield-aware Hierarchical Optimization

Considering process variations in pareto front generation requires statistical simulations for each design point, which typically imposes two orders of magnitude computation cost for statistical performance evaluation. Since there are much more device-level process variables than design variables, scalability and accuracy become crucial to the development of performance models. The more challenging requirement comes from how to perform hierarchical optimization. In the nominal case, circuit blocks can be individually characterized in terms of block-level pareto performance models.

However, in the case of yield-aware optimization, it is critical to capture the impacts of device-level variations on the system-level performances. This need makes individual extraction of block-level pareto models and system-level optimization much more complicated, which are discussed in the following sections.

a. Pareto Front Generation Issues

To obtain yield-aware pareto models, block-level design points that provide robust best block-level performance tradeoffs are collected. This can be achieved by assuming a single yield target for possibly multiple block-level performances, then the pareto front can be generated. However, there exists a disconnection between the block-level models and the system-level performances, where statistical variations are considered at the system level. In other words, the single yield level at the block level is not sufficient to provide enough statistics based on which the whole system yield can be estimated. So a way to generate statistical pareto fronts which can correctly pass yield information as well as block performance trade-offs is needed.

b. System-level Optimization Issues

In [32], the authors suggested to use building block pareto fronts with all the performances at one specified yield level in the hierarchical optimization. And the obtained system-level optimization points were supposed to have the same yield level as building blocks. This approach, however, may not work properly due to several reasons.

Firstly, the transformations from building block-level performances to system-level performances may be complex. This dependency may allow the low yield level of one building block be compensated by other blocks in the same system. Similarly, the performances within one building block may also be compensated by the block-to-system transformation. As a result, the relationship between building block yields

and the system yields can be non-monotonic and complex. Only using the specified yield level pareto fronts in the hierarchical optimization will lose many possible promising block-level performance combinations which may lead to better system-level performances in the end.

The severer problem comes from the statistical correlations between various circuit blocks. In reality, the device variations in different blocks may share common/global physical origins. As a result, not only the device variations are correlated, so are the block-level performances across the blocks. In the prior yield aware pareto front modeling works [33, 32, 10], such correlations are not captured since each block is optimized independently. This issue is especially severe if a single yield target is assumed for all the block-level performances when the pareto models are extracted. Such a *simple* yield-aware pareto model can not provide full statistical information to determine the whole system yield. Consider a simple example, where the entire system consists of two blocks with two block-level performances P_1 and P_2. And the system performance is simply assumed to be: $P_s = P_1 + P_2$. If P_1 and P_2 are of Gaussian distribution then the system P_s will also be Gaussian. In [32], it is suggested that to achieve a system-level yield target, say 84.1%, the block-level pareto models at the same yield level should be considered. In this simple case, the system performance that achieves the yield target is at: $\mu_1 + \mu_2 + \sqrt{\sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2 \cdot cov(P_1, P_2)}$. Obviously, the value for this performance level depends on the correlation between P_1 and P_2. Without such knowledge, the correct system-level performance cannot be decided. The situation becomes even more complex if the block performance distributions are non-Gaussian. In this case, knowing only the correlation factor is also not sufficient. We address these challenges by using the techniques described in the following sections.

3. Multi-yield Pareto Fronts

Unlike the prior work where a single fixed yield level is used when extracting the yield-aware pareto front for multiple performances [33, 32, 10], we introduce the notation of *multi-yield pareto fronts*, where best performance trade-offs are extracted in terms of combinations of yield level parameters individually specified for each performance. For example, a block with two performances P_1 and P_2, will be characterized in terms of two sperate yield level parameters Y_1 and Y_2, one for each performance.

The multi-yield pareto fronts are generated by varying yield level for each performance individually. For the i-th building block our proposed multi-yield pareto front is in the form of

$$\begin{aligned} &MY(\overrightarrow{Y}_{Bi}, \overrightarrow{P}_{Bi}) = 0 \\ &\overrightarrow{Y}_{Bi\_Min} \leq \overrightarrow{Y}_{Bi} \leq \overrightarrow{Y}_{Bi\_Max} \end{aligned} \tag{3.5}$$

where $\overrightarrow{P}_{Bi}$ are the best block performances that can be achieved at the yield level $\overrightarrow{Y}_{Bi}$. For practical purpose, $\overrightarrow{Y}_{Bi}$ is constrained within $[\overrightarrow{Y}_{Bi\_Min}, \overrightarrow{Y}_{Bi\_Max}]$. The yield level $\overrightarrow{Y}_{Bi}$ can vary for different performances in a building block. An example of two-performance multi-yield pareto front generation is shown in Fig. 26. In this case, a fixed-yield pareto model is extracted at each combination of the two performance yield targets.

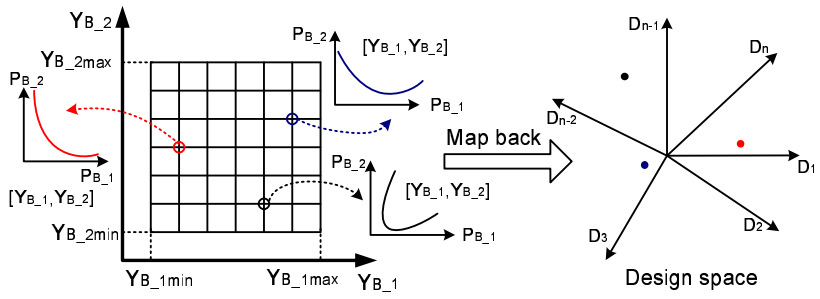

Fig. 26. Multi-yield pareto front generation.

The use of multi-yield pareto fronts allows us to identify a more complete set

of "near optimal″ block-level design points for hierarchical optimization. Facing the lack of system-level interaction during individual pareto model extraction stage, this choice relaxes the artificial constrains set in the fixed-yield pareto models and allows the system-level optimization be conducted in a larger number of promising block level performance tradeoffs. However, the search space for the system-level optimization are still constrained by the block-level best performances trade-offs. Therefore, the hierarchical nature of the overall optimization is preserved.

4. System-level Optimization Formulation

With the multi-yield pareto fronts, we have the information of device level process uncertainties. These multi-yield pareto front models are used to achieve efficient system-level optimization.

a. Bridging Block-level and System-level

In order to evaluate system performance distributions correctly, we need the performance distributions of building blocks for all possible promising design points. As such, the yield levels in multi-yield pareto fronts are not used directly but together with block performances to identify the design parameters. A specified yield level and performance set ($\overrightarrow{Y}_{Bi}$, $\overrightarrow{P}_{Bi}$) can identify a unique design point in the multi-yield pareto front of the i-th building block. When appropriate, interpolation along the pareto front can be conducted. The mapping to the design space of the i-th building block can be denoted as

$$\overrightarrow{D}_{Bi} = D_{PY_i}(\overrightarrow{Y}_{Bi}, \overrightarrow{P}_{Bi}). \tag{3.6}$$

This mapping can be achieved by using Kriging performance models to generate dense points forming pareto fronts instead of analytical formulas, so the mapping back to

the design parameters is naturally obtained.

b. System-level Cost Function

The goal of the yield-aware circuit optimization is to find the optimal system performances at targeted system yields. There can be more than one performances for the whole system. Therefore, we can also specify different system yield levels for different performances. Due to process variations, the system performances are all statistical variables. For the k-th statistical system performance $P_{s,k}$ (smaller the better), suppose we need a yield of $Y_{s,k}$, then the yield-aware performance $P_{s,k}^{Y_{s,k}}$ satisfies the following probability condition

$$\mathrm{P}\{P_{s,k} \leq P_{s,k}^{Y_{s,k}}\} = Y_{s,k}. \tag{3.7}$$

Eqn. 3.7 implies that for $P_{s,k}$, the best achievable performance value is $P_{s,k}^{Y_{s,k}}$ when yield level $Y_{s,k}$ is required. $P_{s,k}^{Y_{s,k}}$ is considered as the yield-aware k-th system performance and to be used in the system-level optimization.

For multi-objective systems, the system-level cost function F for M system performances with yield $Y_s = [Y_{s,1}, \cdots, Y_{s,M}]$ can be formulated as

$$F(\overrightarrow{Y}_B, \overrightarrow{P}_B) = \sum_{k=1}^{M} \frac{W_{s,k} \cdot P_{s,k}^{Y_{s,k}}(\overrightarrow{Y}_B, \overrightarrow{P}_B)}{Spec_k} \tag{3.8}$$

where $W_{s,k}$ is the weighting coefficient for the k-th system performance and $Spec_k$ is its specified performance achievable at yield level $Y_{s,k}$. The input variables for the cost function are the yield and performance set $[\overrightarrow{Y}_B, \overrightarrow{P}_B]$ of all the building blocks. With Eqn. 3.8, the objective of yield-aware optimization is to minimize the cost function $F(\overrightarrow{Y}_B, \overrightarrow{P}_B)$ at specified system yield levels $Y_{s,k}$. By changing the weighting coefficients W_s for different system performances, the system-level optimization can be set to tradeoff between different system performances.

c. Optimization Algorithm

The system cost function has been formulated. Now the question is how to optimize it. As we are interested in the system performances in the statistical sense, accurate evaluation of statistical system performances is required. Since the optimization variables are $[\overrightarrow{Y}_B, \overrightarrow{P}_B]$, we need to know the mapping function $f_{s,k}$ to get $P_{s,k}^{Y_{s,k}}$ from the multi-yield pareto fronts, formulated as $P_{s,k}^{Y_{s,k}} = f_{s,k}(\overrightarrow{Y}_B, \overrightarrow{P}_B)$.

To achieve this, we first transfer the optimization variables in each building block back to the design space using Eqn. 3.6, then we perform Monte-Carlo simulation at each design point to evaluate the block performance distributions. This step can be accelerated by extracting an empirical Kriging based regression model. If there are correlations between device-level parameters across different building blocks, they can be naturally captured in the block performance distributions as it is now possible to generate Monte-Carlo samples at the block level with such correlations considered. The correlated block performance distributions are then mapped into the system performance distributions using system-level behavioral simulation. Again, if needed, this step can be spedup by extracting a Kriging regression model. The system performances at yield Y_s are obtained by finding the values meeting the yield requirement in the system performance distributions. The mapping flow from multi-yield pareto fronts to system performances is shown in Fig. 27.

The flowchart of the hierarchical optimization using multi-yield pareto fronts is illustrated in Fig. 28. We start from the yields and performances of multi-yield pareto fronts, get back to the design points, then obtain the statistical system performances to evaluate the system cost function. The optimization goal is to reduce the cost function in Eqn. 3.8. Since the multi-yield pareto fronts are self constrained, the optimizer also need to take Eqn. 3.6 as the optimization constrain.

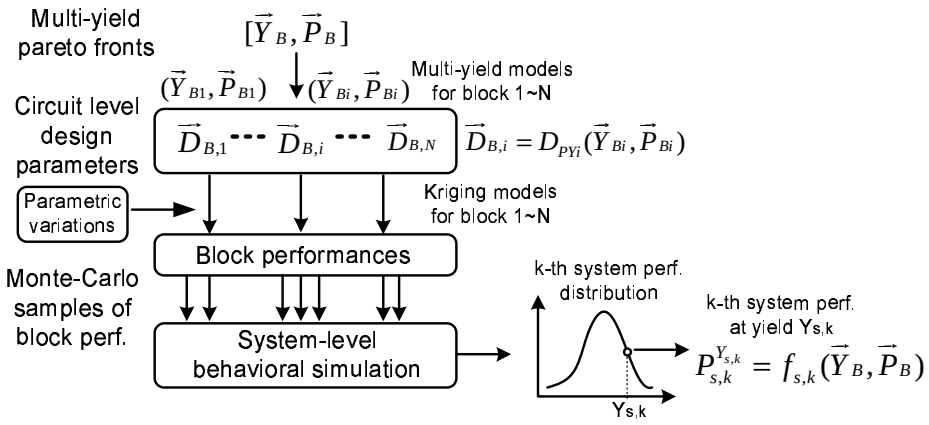

Fig. 27. Mapping from multi-yield pareto fronts to yield-aware system performances.

For a system with M performances and N building blocks, the complete system-level optimization can be formulated as

$$\begin{aligned} & min\ F(\overrightarrow{Y}_B, \overrightarrow{P}_B) \\ & s.t. \begin{cases} P_{s,k}^{Y_{s,k}} = f_{s,k}(\overrightarrow{Y}_B, \overrightarrow{P}_B), & k = 1, 2, \cdots M \\ MY(\overrightarrow{Y}_{Bi}, \overrightarrow{P}_{Bi}) = 0, & i = 1, 2, \cdots N \\ \overrightarrow{Y}_{Bi\_Min} \leq \overrightarrow{Y}_{Bi} \leq \overrightarrow{Y}_{Bi\_Max} \end{cases} \end{aligned} \tag{3.9}$$

where the optimization variables are multi-yield pareto front yields and performances $[\overrightarrow{Y}_B, \overrightarrow{P}_B]$ for all the N building blocks. The dimensions of $\overrightarrow{Y}_{Bi}$ and $\overrightarrow{P}_{Bi}$ in each building block are the same, which depend on the number of block-level performances considered. Note that each multi-yield pareto front acts as an constraint in the optimization and reduces the degrees of freedom for of the system-level optimization by one. Suppose the i-th block of has L_i block-level performances, the dimension of the optimization search space is $\sum_i^N (2L_i - 1)$. Hence, the proposed approach extends the efficiency of hierarchical analog optimization from deterministic optimization to statistical optimization.

The system-level optimization problem can be solved by any suitable optimization method, particularly a derivative free method. Global optimization algorithm

based on multilevel coordinate search (MCS) [34] is adopted in the optimization flow.

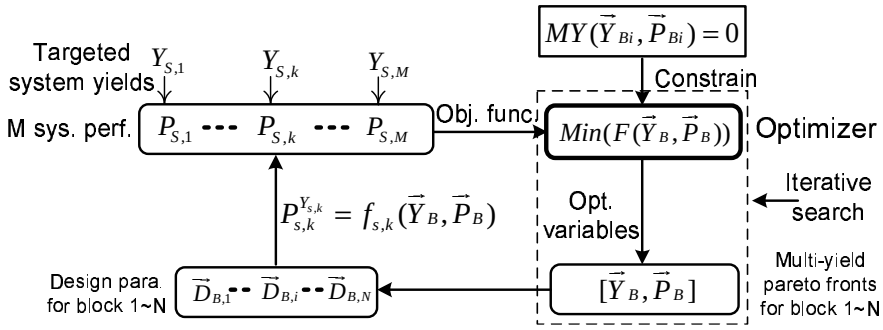

Fig. 28. Hierarchical optimization using multi-yield pareto fronts.

We summarize the complete algorithm consisting of Kriging modeling construction, multi-yield pareto front generation and system-level optimization in Algorithm 2.

Algorithm 2 Yield-aware Hierarchical Optimization Flow

Inputs: Design variable set $\overrightarrow{D} = \{D_1, D_2, \cdots, D_n\}$, system specifications $\{Spec_1, Spec_2, \cdots, Spec_m\}$ and weighting coefficients $\{\omega_1, \omega_2, \cdots, \omega_m\}$

Outputs: Optimized system performances $\overrightarrow{P}_{sysopt}$ and corresponding design variables $\overrightarrow{D}_{opt}$.

1: divide system into N sub-blocks
2: construct Kriging performance model $\overrightarrow{K}(\overrightarrow{D})$ in nominal case for each building block
3: generate nominal pareto front $\overrightarrow{PF}(\overrightarrow{P}_B)$ using iterative search
4: construct Kriging performance model $\overrightarrow{K}(\overrightarrow{D}, \overrightarrow{V})$ containing both design and process variables
5: build multi-yield pareto front $MY(\overrightarrow{Y}_{Bi}, \overrightarrow{P}_{Bi}) = 0$ for each building block
6: construct system cost function using Eqn. 3.8
7: $\overrightarrow{P}_{sysopt} = min(F(\overrightarrow{Y}_B, \overrightarrow{P}_B))$ *s.t.* Eqn. 3.9
8: return $\overrightarrow{P}_{sysopt}$ and $\overrightarrow{D}_{opt}$

5. System Optimization Examples

We demonstrate the detailed applications of the proposed yield-aware hierarchical optimization including behavioral modeling, multi-yield pareto front generation and system-level optimization formulation by looking into two design examples in this section.

a. Two-stage Amplifier

To illustrate important aspects of yield-aware hierarchical optimization, we use a simple but revealing two-stage operational amplifier example as shown in Fig. 29. This amplifier is designed for the application of low-bandwidth pre-amplification and requires low-power consumption with reasonable gain. The amplifier is implemented in a 90nm CMOS technology. The design parameters consist of transistor sizes, biasing currents and the capacitance. We consider a global process variable of gate oxide thickness Tox with 3σ=15% for all the transistors, 3σ=10% mismatch of gate length L for each transistor and 3σ=10% variation for capacitance and resistance.

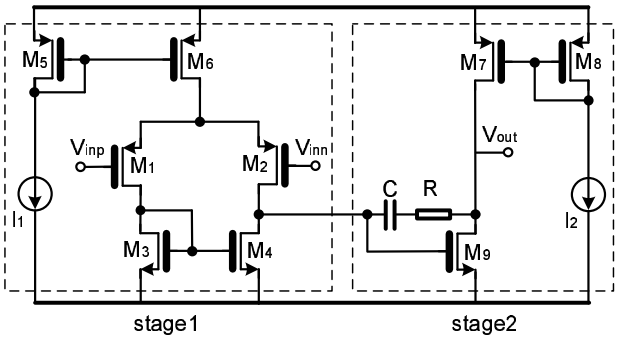

Fig. 29. Schematic of two-stage operational amplifier.

The amplifier is partitioned into two stages and we model each stage as one building block. The global process variable Tox are modeled as input variable in the Kriging performance models of both stages. The block performances are gain

and power, the system performances selected are also gain and power. We also impose a requirements of phase margin to make sure the optimized circuits are stable. For this circuit, the behavioral models that transfer block performances to system performances are rather straightforward and can be evaluated analytically:

$$\begin{aligned} Gain_{system} &= Gain_{stage1} \times Gain_{stage2} \\ Power_{system} &= Power_{stage1} + Power_{stage2} \end{aligned} \tag{3.10}$$

The cost function in Eqn. 3.8 is rewritten for the amplifier case,

$$F(\overrightarrow{Y}_B, \overrightarrow{P}_B) = W_G \cdot \frac{Gain_{amp}(Y_G)}{Gain_{Spec}} + W_P \cdot \frac{Pow_{amp}(Y_P)}{Pow_{Spec}} \tag{3.11}$$

The optimization target is to reduce the system cost function together with the phase margin requirement, here we set the phase margin to be larger than 60^o, so the system optimization is formulated as

$$\begin{aligned} &\min \left(F(\overrightarrow{Y}_B, \overrightarrow{P}_B) \right) \\ &s.t. \quad PM \geq 60^o \end{aligned} \tag{3.12}$$

In order to verify if the multi-yield hierarchical optimization can find the actual system optimal point, we compare the results of the proposed method and that of the flat yield-aware optimization method. The fixed-yield hierarchical optimization method is also evaluated for comparison. The flat optimization is performed by using a simulation-based optimization approach similar to a commercial optimization tool [28]. The optimizer [34] calls Spectre [11] to run Monte-Carlo simulation at each design point it reaches and uses the simulated performances at targeted yields level as the guidance for brute-force optimization search. The major computation cost for the proposed multi-yield hierarchical optimization method comes from generating the multi-yield pareto fronts, which needs around 20 minutes. Once the multi-yield pareto fronts are obtained, system-level optimization requires only 1-2 minutes, since the

performance mapping from block-level to the system-level is analytical as in Eqn. 3.10.

The comparison of results of different optimization methods are illustrated in Fig. 30. The system-level yields Y_G and Y_P are set to 70% for both gain and power. The data in the plots are all evaluated with transistor-level Monte-Carlo simulation using Spectre for better accuracy. From the figure, we can see that the system trade-offs at the converged optimization points captured by the proposed method match the flat optimization results very well, while the fixed yield pareto front optimization is not able to converge to the actual optimal system performance trade-off curves.

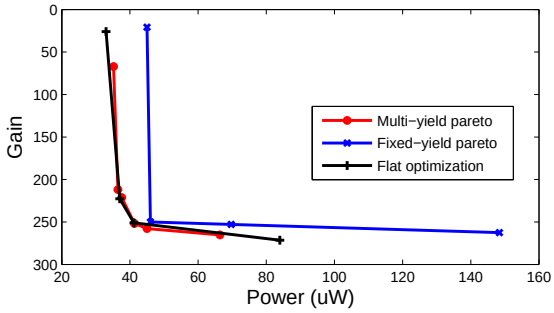

Fig. 30. Comparison of results of different optimization methods.

From the amplifier example presented above, we can see clearly that directly applying the pareto fronts at fixed yield levels in the hierarchical optimization is not able to find the system optimal solutions, while the results of proposed optimization methodology with multi-yield pareto fronts match the flat optimization results very well. Although the circuit by itself is simple, it demonstrates that the conventional hierarchical yield-aware optimization method is not suitable and our proposed multi-yield pareto optimization approach can be a good solution to perform yield-aware hierarchical optimization. Next we will discuss a more realistic and complicated case, where flat optimization is not possible because of the huge simulation time needed for the direct simulation.

b. Charge-pump PLL

Due to the nature of coexistence of fast and slow signals, the design and evaluation of PLL system is quite complex and costly. So the brute-force optimization by searching in the design space with transistor-level simulation is infeasible for PLL designs. In this example, we demonstrate the effectiveness of the proposed yield-aware optimization methodology using a charge-pump PLL optimization as design example.

The PLL system investigated contains a voltage control oscillator (VCO), a phase detector, a charge pump, a loop filter and a frequency divider. Among these components, the phase detector and the frequency divider are digital components which have few tunabilities and are robust to process variations. So we restrict our focus on the optimization of VCO, charge pump and filter in this example.

The performances selected to build VCO behavioral models are *jitter*, *power*, VCO gain K_{VCO} and Frequency F_{offset} at certain control voltage V_{offset}. The VCO voltage-frequency curve is linearized around V_{offset} with the slope of K_{VCO}. V_{offset} is predefined as the center of VCO linear gain region. These coefficients are used to capture the complete voltage-frequency curve, which preciously models the PLL acquisition procedure especially in low/high control voltages. The model used is of advantage to the conventional models only considering the VCO gain within a fixed frequency region [15, 35], since the VCO behaviors in nonlinear regions are also captured here.

For the charge pump, we include *jitter*, charge up current I_{up} and charge down current I_{down} in the behavioral model. The parameters of loop filters are the capacitances of C_1 and C_2 with the resistance of R, which can all be handled directly at system-level simulation. The power of charge pump and loop filter can be calculated with charge pump currents [15]. The behaviors of digital building blocks in the sys-

tem, including frequency divider and the phase detector, are also characterized using delay and slew and included in the behavioral models. The simulation and modeling of PLL is shown in Fig.31.

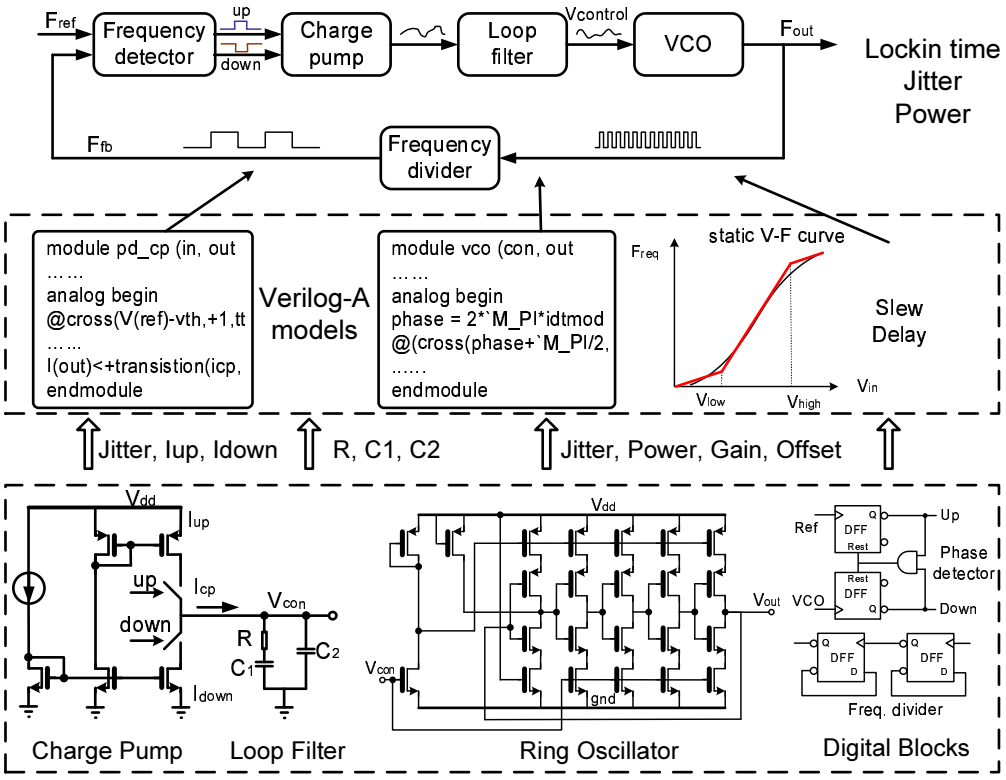

Fig. 31. PLL modeling and optimization.

When the behaviors of analog building blocks are extracted, they are mapped into Verilog-A models for the system-level simulation [15, 36]. In the system level, the performances of PLL are considered as lockin time T, power P (exclude digital blocks) as well as jitter J [37]. With the Verilog-A behavioral models, the mapping from block-level performances to the system-level performances can be achieved efficiently.

Multi-yield pareto fronts for individual building block are extract to perform yield-aware hierarchical optimization. For VCO, since the frequency F_{offset} is only a model parameter which is not considered as performance metric, we generate the

pareto fronts considering power, jitter and gain as $P_{VCO}(power, jitter, gain)$. The designed I_{up} and I_{down} should be equal for the charge pump, but when the process variations are considered, there will be mismatch between I_{up} and I_{down}, which can introduce extra jitter and impact lockin time significantly. So for the charge pump, we generate pareto fronts considering jitter, average charge current $I_{cp} = 0.5 \cdot (I_{up} + I_{down})$ and mismatch current $I_{mis} = abs(I_{up} - I_{down})$ as $P_{CP}(jitter, I_{cp}, I_{mis})$. Charge pump power is not included in the pareto fronts since it is proportional to I_{cp}. The cost function in Eqn. 3.8 is rewritten for the PLL case as

$$F(\overrightarrow{Y}_B, \overrightarrow{P}_B) = W_P \cdot \frac{P_{PLL}(Y_P)}{P_{Spec}} + W_J \cdot \frac{J_{PLL}(Y_J)}{J_{Spec}} + W_T \cdot \frac{T_{PLL}(Y_T)}{T_{Spec}} \tag{3.13}$$

The design space constrains in Eqn. 3.9 is further expressed for the PLL case as

$$\begin{cases} MY_{CP}(Y_{jit}, P_{jit}, Y_{Icp}, P_{Icp}, Y_{Imis}, P_{Imis}) = 0 \\ MY_{VCO}(Y_{jit}, P_{jit}, Y_{pow}, P_{pow}, Y_{gain}, P_{gain}) = 0 \\ R_{min} \leq R \leq R_{max}; C_{min} \leq C \leq C_{max}; Y_{min} \leq Y \leq Y_{max} \end{cases} \tag{3.14}$$

The input variables for the system level optimization include not only the yield-levels and performances of VCO and charge pump multi-yield pareto fronts, but also loop filter parameters since they can be evaluated directly in the system-level simulation.

The PLL example is implemented in 90nm CMOS technology, the process variations considered include threshold voltage V_{th} for each transistor with a variation of 3σ=10%. For the system-level simulation, a single transient simulation of $10\mu s$ for jitter and lockin time analysis requires about 40 seconds even with Verilog-A models. To alleviate the optimization cost, we build another Kriging model to map all the building block-level performances (VCO_{power}, VCO_{jitter}, etc) and filter parameters to the PLL system performances, then use these high-level Kriging models to guide the optimizer to find optimal solutions. To ensure the accuracy, when the optimiza-

tion design points are obtained, we use Spectre and Verilog-A models to find the performances with specified yields in these points.

The examples of multi-yield pareto fronts for charge pump and VCO are illustrated in Fig. 32.

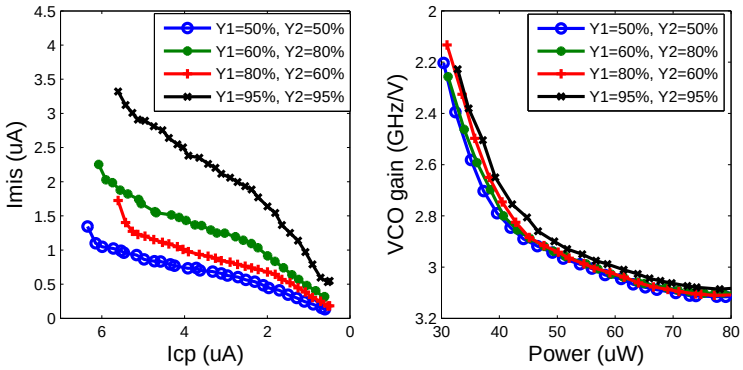

Fig. 32. Multi-yield pareto fronts for charge pump (left) and VCO (right).

In the system level, the lockin time and power trade-offs for the PLL are analyzed. We change the weighting coefficients for power and lockin time in Eqn. 3.13, then different optimal PLL performance combinations can be obtained, as shown in Fig. 33. The two system performances are set to the same yield level. The system performances shown in the figure are simulated using Spectre with Verilog-A models. The performance trade-offs without consideration of yield are obtained similar to [15], also shown in Fig. 33. We can see from the figure that the yield could be very low if we do the optimization without considering the yield information, especially for the region around the point denoted as "Opt. point". The nominal system performances of "Opt. point" and the initial design point evaluated using direct Spectre simulation are also plotted in Fig. 33, denoted as "Init. design" and "Opt. design", respectively.

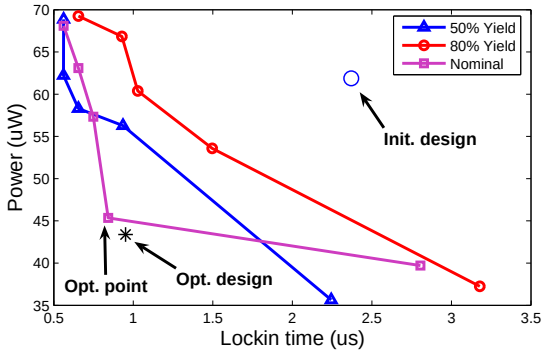

Fig. 33. Trade-offs of lockin time and power at different yield levels.

Table VI. Hierarchical optimization results for PLL.

	PLL		VCO				Charge pump		
	Power	Lockin	Gain	Offset	Power	Jitter	I_{up}	I_{down}	Jitter
Init. D	61.9μW	2.37μs	2.43G/V	1.54GHz	57.2μW	0.87ps	2.11μA	2.07μA	54.2fs
Opt. D	43.4μW	0.95μs	2.36G/V	1.36GHz	36.0μW	0.92ps	3.55μA	3.53μA	63.0fs
Opt. P	45.4μW	0.84μs	2.39G/V	1.33GHz	36.7μW	1.18ps	3.64μA	3.62μA	68.6fs

Table VI shows the results of nominal performances comparison for the initial design and the "Opt. point" in the obtained pareto front. The system and block-level performances of initial design and "Opt. point" are both evaluated with direct transistor-level Spectre simulation and listed in the first and second row, respectively. The PLL system results in the third row are simulated using Spectre and Verilog-A models, the building block performances are obtained from Kriging models for the design parameters of "Opt. point". We can see they are quite close to the direct Spectre simulation as in the second row, which validates the accuracy of our Verilog-A behavioral models.

Similarly, the trade-offs between jitter and lockin time at different yield levels are plotted in Fig. 34.

As stated before, it is not possible to use direct transistor-level optimization to verify if the proposed method can find the optimal system-level performance trade-offs

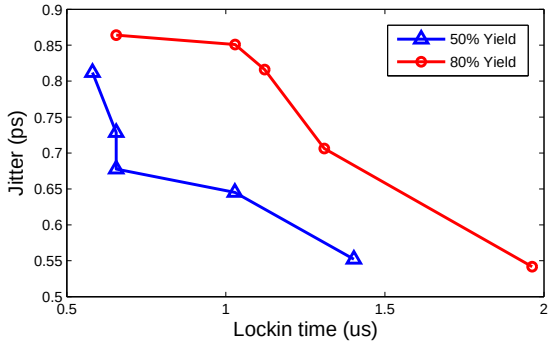

Fig. 34. Trade-offs of lockin time and jitter at different yield levels.

as in the two-stage amplifier case. So we randomly sample in the design space and simulate the corresponding PLL system performances using the Verilog-A models. Fig. 35 shows an example of the trade-offs of power and jitter at 80% yield level and the performances of 200 random design samples. It can be seen that the performance trade-off curve in Fig. 35 is superior to all the verification points, which confirms that our proposed method achieves optimal designs.

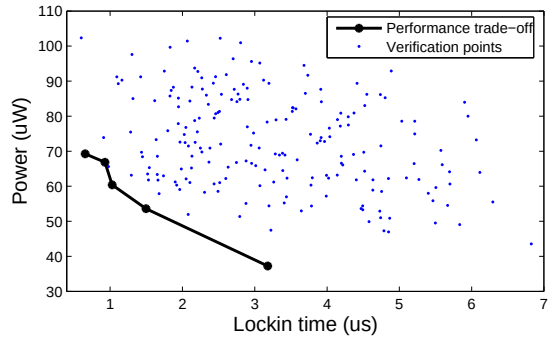

Fig. 35. Verification of performance trade-offs for lockin time and power.

The runtime information is summarized in Table VII, the computation cost for the complete flow is only a few hours, which is quite acceptable for large analog circuit synthesis.

Table VII. Runtime summary for PLL optimization.

	VCO	Charge pump
Kriging model gen.	12 min	5 min
Multi-yield pareto gen.	103 min	117 min
Cost function optimization	10 min/point	

C. Summary

Automated syntheses of complex analog systems in scaled CMOS technologies are proposed by conducting numerical optimization based on efficient circuit modeling and system optimization formulation in this chapter. We start with yield-aware optimization for small-scale analog circuits by employing iterative search based optimization approach to efficiently seek optimal performance tradeoffs under yield constraints in high-dimensional design parameter and process variation spaces. We demonstrate the proposed approach by constructing pareto front analog performance models. Moreover, specific techniques including partial Kriging evaluation approach have been developed to facilitate efficient design space exploration while considering process variations. Experimental results of one oscillator and one operational amplifier confirm the good accuracy and efficiency of the presented optimization approach.

In order to handle the large analog/mixed-signal system optimization problems, we further propose a methodology to perform yield-aware hierarchical optimization by addressing the problems of generating yield-aware pareto fronts of building blocks and formulating system performances at specified yield levels. The system-level performance distributions are naturally captured by searching in the multi-yield pareto fronts while still maintaining the efficiency of hierarchical optimization. The proposed methodology is validated through the optimizations of a two-stage amplifier and a

large charge-pump PLL design with good efficiency achieved in both design cases.

CHAPTER IV

ON-CHIP TEST FOR ANALOG/MIXED-SIGNAL CIRCUITS

Besides performing yield-aware system optimization, the other approach to alleviate the influences of process variations in analog/mixed-signal systems is to use on-chip test and performance compensation functions to self heal the failing chips caused by process variations. The implementations of on-chip test/diagnoise idea can vary. The access of internal analog signals could be very difficult for integrated circuits, so circuitries are designed to modify the original structures to improve the circuit performance accessibilities. These design techniques that add testability features are classified as design for test (DFT), which make the testing of analog/mixed-signal circuit easier. A more complete realization is to integrate the measurement and failure classification function blocks in the analog/mixed-signal circuits so the systems can perform self diagnose and make go/no-go decisions. This approach is called built-in self-test (BIST) [38].

Careful design and optimization for on-chip testing function blocks are required for efficient and effective parametrical yield capturing. In this chapter we first propose a cost-effective linearity test method targeting for switched-capacitor Sigma-Delta ADCs [39]. The underlining system analysis supports our idea that complex system performances can be indirectly tested by some easy-to-measure alternatives. As such we can achieve efficient and accurate built-in self-testing in analog/mixed-signal systems. We further look into different design choices of design-for-test schemes for charge-pump PLLs and perform optimization to enhance test circuit efficiency [9].

A. Linearity Test for Sigma-Delta ADCs

Static linearity test of Sigma-Delta ADCs imposes stringent requirement on the precision of test signals and leads to excessive test time. Consequently, ADC test remains as a bottleneck to the product development and contributes significantly to the devolvement cost. In this section, a cost-effective linearity test and diagnosis methodology is presented for Sigma-Delta ADCs with multi-bit internal DACs. Frequency-domain nonlinear circuit analysis is employed to systematically establish the connection between the static linearity measure (INL) and its frequency domain counterpart (harmonic distortions (HDs)), making it possible to predict INL using much simpler HD measurements. The efficacy of the proposed technique is demonstrated by successful construction of accurate simulation-based INL prediction models which are also compared against with closed-form models resulted directly from our circuit analysis.

1. System Analysis Using Volterra Series

Our central approach in achieving the low-cost ADC test and diagnosis is to predict INL via frequency-domain measurements (HDs) easily obtainable by applying a sinusoidal input to the circuit under test (CUT). To achieve this goal, the connection between INL and HDs must be established. This is accomplished by performing discrete-time Volterra series analysis suitable for analyzing the nonlinear circuit behaviors of the targeted switched-capacitor $\Sigma\Delta$ ADCs.

In Volterra series, the output of a nonlinear system is considered as a sum of the responses of increasing orders [40]. In the frequency domain, Volterra series can be used rather straightforwardly to calculate harmonic distortions and intermodulations under multiple-tone excitations. If the input signal $u(k)$ is an n-tone input $u(k) =$

$e^{j\omega_1 k} + e^{j\omega_2 k} + \cdots + e^{j\omega_n k}$, the intermodulation component at the sum frequency $\omega_1 + \omega_2, \cdots, \omega_n$ can be written as

$$y_n(k) = n! H_n(\omega_1, \omega_2, \cdots, \omega_n) e^{jk(\omega_1 + \omega_2, \cdots + \omega_n)} \tag{4.1}$$

where $H_n(\omega_1, \omega_2, \cdots, \omega_n)$ is the n-th order nonlinear transfer function. The nonlinear transfer functions can be considered as extensions to the familiar linear (first order) transfer function and are used as a canonical characterization of the weakly nonlinear system behavior. For $\Sigma\Delta$ ADCs, the transfer functions can be used to calculate HDs and INLs.

a. Nonlinear System Modeling

The switched-capacitor $\Sigma\Delta$ ADCs are intrinsically nonlinear discrete-time systems. To make Volterra series applicable for $\Sigma\Delta$ ADCs, we need to model the internal quantizers properly. A widely used approach in design analysis is to consider the quantizer as a linear gain model with added quantization noise at the output [4]. Usually, this additive quantization noise can be modeled as white noise over the entire signal bandwidth. In spite of the approximation introduced, this choice allows us to consider the strong nonlinear behavior of quantization through the standard means of quantization noise.

For each switched-capacitor integrator in the system, the state transfer can be modeled as a nonlinear function of the integrator's inputs and its current state

$$y(k+1) = F(y(k), x(k), d(k)), \tag{4.2}$$

where $y(k+1)$ is the current output of the integrator, $y(k)$ is the previous output, $x(k)$ and $d(k)$ are the previous input signal and digital feedback signal, respectively.

The nonlinear function can be further written as

$$F(y(k), x(k), d(k)) = y(k) + a_1 \cdot u(k) + a_2 \cdot u(k)^2 + \cdots + a_n \cdot u(k)^n \tag{4.3}$$

where $u(k)$ is the difference between the input analog signal and the digital feedback signal, a_1 is the linear gain of the integrator transfer characteristics, and $a_2, \cdots, a_n$ are the second-order to the n-th order coefficients which are used to model the nonlinearities of the integrator.

The transfer curve of an internal D/A converter in the system can be written as

$$f = b_0 + b_1 \cdot D + b_2 \cdot D^2 + \cdots + b_n \cdot D^n \tag{4.4}$$

where f is the DAC output, D is the digital input, b_0 is the offset, $b_1 \cdot D, \cdots, b_n \cdot D^n$ are the first to the n-th order components at the DAC output.

b. Nonlinear Transfer Function Analysis

With the previous analysis, we start to derive transfer functions in the frequency domain using Volterra series. As an example, we will consider a second-order ΣΔ ADC with 2-bit internal quantizer as shown in Fig. 36.

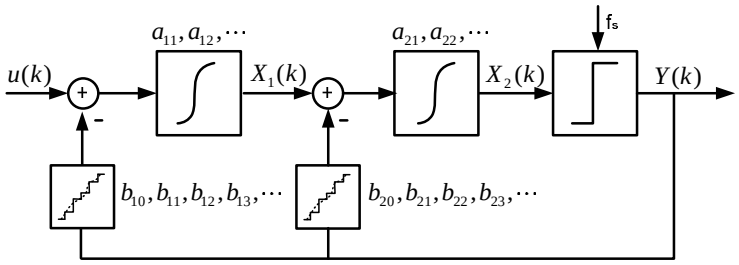

Fig. 36. Modeling of a second-order ΣΔ ADC.

The nonlinear characteristics of the integrators and the internal DAC will be modeled using second and third order polynomials, respectively. The state equations

for the whole system can be written as

$$\begin{aligned} X_1(k+1) &= X_1(k) + a_{11}F_{d,1}(k) + a_{12}F_{d,1}(k)^2, \\ X_2(k+1) &= X_2(k) + a_{21}F_{d,2}(k) + a_{22}F_{d,2}(k)^2, \\ Y(k) &= \alpha \cdot X_2(k), \end{aligned} \tag{4.5}$$

where $X_1(k)$ is the output of the first-stage integrator, $X_2(k)$ is the output of the second-stage integrator and $Y(k)$ is the digital output. $F_{d,1}(k)$ and $F_{d,2}$ are given as

$$\begin{aligned} F_{d,1}(k) &= u(k) - b_{10} - b_{11}Y(k) - b_{12}Y(k)^2 - b_{13}Y(k)^3 \\ F_{d,2}(k) &= X_1(k) - b_{20} - b_{21}Y(k) - b_{22}Y(k)^2 - b_{23}Y(k)^3, \end{aligned} \tag{4.6}$$

where $b_{10}, b_{11}, b_{12}, b_{13}, b_{20}, b_{21}, b_{22}, b_{23}$ are the coefficients of transfer curves for the two internal feedback DACs, a_{11}, a_{12}, a_{21}, a_{22} are the nonlinear coefficients for the two integrators and α is the linear gain for the quantizer. In Volterra analysis, transfer functions and responses at different orders are analyzed recursively [41, 42]. We start from deriving the first order (linear) transfer functions. When a single-tone signal $e^{j\omega k}$ is applied as the input, we substitute each output $X(k)$ by its linear response $X(k) = H_1(\omega)e^{j\omega k}$ into Equation 4.5 and solve the resulting system equations. We obtain the first order transfer functions as

$$\begin{aligned} H_1^{x_1}(\omega) &= \frac{a_{11}(e^{j\omega}-1)+a_{11}a_{21}b_{21}\alpha}{(e^{j\omega}-1)^2+a_{21}b_{21}\alpha(e^{j\omega}-1)+a_{11}a_{21}b_{11}\alpha}, \\ H_1^{x_2}(\omega) &= \frac{a_{11}a_{21}}{(e^{j\omega}-1)^2+a_{21}b_{21}\alpha(e^{j\omega}-1)+a_{11}a_{21}b_{11}\alpha}, \\ H_1^{out}(\omega) &= \frac{a_{11}a_{21}\alpha}{(e^{j\omega}-1)^2+a_{21}b_{21}\alpha(e^{j\omega}-1)+a_{11}a_{21}b_{11}\alpha}, \end{aligned} \tag{4.7}$$

where $H_1^{out}(\omega)$ is the first order transfer function at the quantizer output.

To derive the second order transfer functions, a two-tone input $e^{j\omega_1 k} + e^{j\omega_2 k}$ is applied to the system. For any response $X(k)$, we consider its frequency component

at $\omega_1 + \omega_2$, which is given as

$$X(k) = 2H_2(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k}. \tag{4.8}$$

We substitute each second order response into Equation 4.5 and keep only the signal components at $\omega_1 + \omega_2$. A set of equations are obtained as

$$\begin{aligned}
&2H_2^{x_1}(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)(k+1)} = 2H_2^{x_1}(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k} \\
&+a_{11}[-2b_{11}Y(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k} - 2b_{12}H_1(\omega_1)H_1(\omega_2)e^{j(\omega_1+\omega_2)k}] \\
&+a_{12}[-b_{11}H_1(\omega_1) - b_{11}H_1(\omega_2)]^2, \\
\\
&2H_2^{x_2}(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)(k+1)} = 2H_2^{x_2}(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k} \\
&+a_{21}[2H_2^{x_1}(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k} - 2b_{21}Y(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k} \\
&-2b_{22}H_1^{x_2}(\omega_1)H_1^{x_2}(\omega_2)e^{j(\omega_1+\omega_2)k}] \\
&+a_{22}[-b_{21}H_1(\omega_1)e^{j\omega_1 k} - b_{21}H_1(\omega_2)e^{j\omega_2 k}]^2, \\
\\
&Y(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k} = \alpha H_2^{x_2}(\omega_1, \omega_2)e^{j(\omega_1+\omega_2)k},
\end{aligned} \tag{4.9}$$

where $H_2^{x_1}(\omega_1, \omega_2)$ and $H_2^{x_2}(\omega_1, \omega_2)$ are the second order transfer functions at the first and the second stage integrator outputs. The second order transfer function of the ADC output $H_2^{out}(\omega_1, \omega_2)$ can be shown to be

$$\begin{aligned}
&H_2^{out}(\omega_1, \omega_2) = \alpha\tfrac{F_{21}}{F_{22}}H_1(\omega_1)H_1(\omega_2), \\
&F_{21} = 2a_{12}b_{12} - a_{11}a_{21}b_{12} - (2a_{22}b_{22}^2 - a_{21}b_{22})(e^{j(\omega_1+\omega_2)} - 1), \\
&F_{22} = (e^{j(\omega_1+\omega_2)} - 1)^2 + a_{21}b_{21}\alpha(e^{j(\omega_1+\omega_2)} - 1) + a_{11}a_{21}b_{11}\alpha,
\end{aligned} \tag{4.10}$$

In a similar way, the third order transfer function of the ADC output can be

derived as

$$
\begin{aligned}
&H_3^{out}(\omega_1,\omega_2,\omega_3) = \alpha\tfrac{F_{31}}{F_{33}} + \alpha\tfrac{F_{32}}{F_{33}}(e^{j(\omega_1+\omega_2+\omega_3)}-1),\\
&F_{31} = (2a_{21}a_{12}b_{11}^2 - 2a_{21}a_{11}b_{12})H_1(\omega_1)H_2(\omega_2,\omega_3)\\
&+(2a_{21}a_{12}b_{11}b_{12} - a_{21}a_{11}b_{13})H_1(\omega_1)H_1(\omega_2)H_1(\omega_3),\\
&F_{32} = (2a_{22}b_{21}^2 - 2a_{21}b_{12})H_1(\omega_1)H_2(\omega_2,\omega_3)\\
&+(2a_{22}b_{11}b_{12} - 2a_{21}b_{12})H_1(\omega_1)H_1(\omega_2)H_1(\omega_3),\\
&F_{33} = (e^{j(\omega_1+\omega_2+\omega_3)}-1)^2 + a_{21}b_{21}\alpha(e^{j(\omega_1+\omega_2+\omega_3)}-1)\\
&+a_{11}a_{21}b_{11}\alpha.
\end{aligned}
\tag{4.11}
$$

We only consider transfer functions up to the third order, which are usually adequate to describe the weakly nonlinear system behavior.

2. Predicting INL using HDs

a. Relating INL with Transfer Functions

The definition of INL is the deviation of the actual ADC transfer curve from the ideal transfer curve. The ideal curve can be determined either by a least-square fitting through the acquired samples or by a straight line through the two end points of the ADC [43]. Here, the two end points method is used to define the INL [44] as shown in Fig. 37.

The ADC transfer curve after offset and gain correction can be represented by an n-th order polynomial

$$y_{actual}(x) = \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n, \tag{4.12}$$

where $y_{actual}(x)$ is the ADC output, x is the analog input and $\beta_1, \beta_2, \cdots, \beta_n$ are the coefficients to model the system nonlinearity. Suppose that the input ramp signal used to measure the INL rises from $-A$ to A. The slope γ of the ideal transfer curve connecting the two end points can be determined by evaluating the two end point

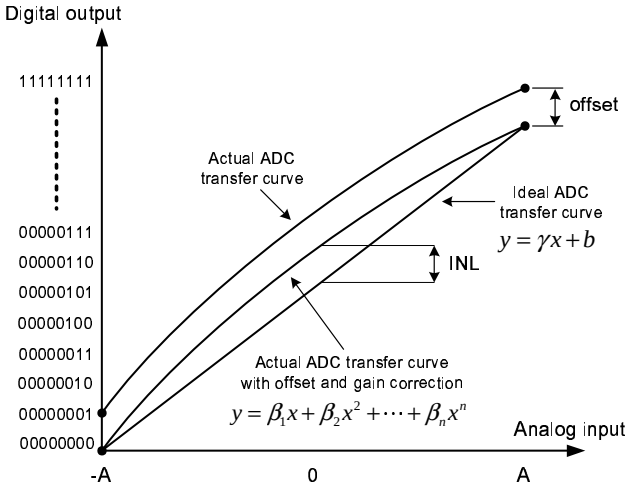

Fig. 37. Definition of integral nonlinearity.

ADC outputs using nonlinear transfer functions at DC

$$\gamma = \frac{AH_1(0)+\cdots+A^nH_n(0,\cdots,0)}{A-(-A)} - \frac{(-A)H_1(0)+\cdots+(-A)^nH_n(0,\cdots,0)}{A-(-A)} \tag{4.13}$$

If only the first three orders of transfer functions are considered, the ideal transfer slope becomes $H_1 + A^2 \cdot H_3$ with offset $A^2 \cdot H_2$. So the ideal transfer curve can be written as

$$y_{ideal}(x) = (H_1 + A^2 \cdot H_3) \cdot x + A^2 \cdot H_2 \tag{4.14}$$

where H_1, H_2, H_3 are the transfer functions at DC. With Equation 4.12 and Equation 4.14, INL can be written as

$$\begin{aligned} INL(x) &= V_{scale} \cdot (x^2H_2 + x^3H_3 - x \cdot A^2 \cdot H_3 - A^2 \cdot H_2) \\ V_{scale} &= \frac{2^n-1}{2\mathrm{A}\cdot(H_1+A^2\cdot H_3)} \end{aligned}, \tag{4.15}$$

where V_{scale} is the scaling factor used to convert the deviation from the ideal transfer curve into LSB scale.

b. Relating INL with HDs

The relationship between INL and transfer functions have been derived in the previous section. Measuring harmonic distortions using automatic test equipment will become a feasible way to estimate INL if a relationship of INL and HDs can be derived. Consider applying a sinusoidal input $A \cdot cos(\omega k)$ to measure the HDs. Such input can be decomposed into a pair of two complex exponentials $A/2 \cdot e^{j\omega k}$ and $A/2 \cdot e^{-j\omega k}$. The corresponding response can be computed using nonlinear transfer functions and leads to the following expressions for HDs

$$\begin{aligned}
l_1(\omega) &= \tfrac{1}{2}AH_1(\omega)e^{j\omega k} + \tfrac{1}{2}AH_1(-\omega)e^{-j\omega k} \\
&= \tfrac{1}{2}AH_1(\omega)e^{j\omega k} + \tfrac{1}{2}AH_1^*(\omega)e^{-j\omega k}, \\
l_2(2\omega) &= \tfrac{1}{4}A^2H_2(\omega,\omega)e^{2j\omega k} + \tfrac{1}{4}A^2H_2(-\omega,-\omega)e^{-2j\omega k} \\
&= \tfrac{1}{4}A^2H_2(\omega,\omega)e^{2j\omega k} + \tfrac{1}{4}A^2H_2^*(\omega,\omega)e^{-2j\omega k}, \\
l_3(3\omega) &= \tfrac{1}{8}A^3H_3(\omega,\omega,\omega)e^{3j\omega k} + \tfrac{1}{8}A^3H_3(-\omega,-\omega,-\omega)e^{-3j\omega k} \\
&= \tfrac{1}{8}A^3H_3(\omega,\omega,\omega)e^{3j\omega k} + \tfrac{1}{8}A^3H_3^*(\omega,\omega,\omega)e^{-3j\omega k},
\end{aligned} \tag{4.16}$$

where $l_1(\omega)$, $l_2(2\omega)$ and $l_3(3\omega)$ are the first, second, third order harmonic components at the output, H_1, H_2, H_3 are the first, second, third order transfer functions, and H_1^*, H_2^*, H_3^* are the conjugates of H_1, H_2, H_3, respectively. Combining the real and imaginary parts of the response, we can rewrite Equation 4.16 as

$$\begin{aligned}
l_1(\omega) &= A\mathrm{Re}(H_1(\omega))\cos(\omega k) - A\mathrm{Im}(H_1(\omega))\sin(\omega k), \\
l_2(2\omega) &= \tfrac{1}{2}A^2\mathrm{Re}(H_2(\omega,\omega))\cos(2\omega k) - \tfrac{1}{2}A^2\mathrm{Im}(H_2(\omega,\omega))\sin(2\omega k), \\
l_3(3\omega) &= \tfrac{1}{4}A^3\mathrm{Re}(H_3(\omega,\omega,\omega))\cos(3\omega k) \\
&\quad -\tfrac{1}{4}A^3\mathrm{Im}(H_3(\omega,\omega,\omega))\sin(3\omega k).
\end{aligned} \tag{4.17}$$

Notice that one of the characteristics of $\Sigma\Delta$ ADCs is oversampling. This means the sampling frequency is much higher than the input signal bandwidth. The maximum

discrete-time input signal frequency is given as

$$\omega_{max} = 2\pi \cdot f_b / f_{sample} = \pi / OSR, \tag{4.18}$$

where f_b is the input signal bandwidth, f_{sample} is the sampling frequency and OSR is the oversampling ratio. Normally OSR is quite large, so ω_{max} is very small and the imaginary parts of transfer functions in Equation 4.11 can be neglected. Therefore, the real parts of the transfer functions are good approximation of the complete transfer functions. In this case, Equation 4.16 can be rewritten as

$$\begin{aligned} l_1(\omega) &\approx AH_1(\omega), \\ l_2(2\omega) &\approx \tfrac{1}{2}A^2H_2(\omega,\omega), \\ l_3(3\omega) &\approx \tfrac{1}{4}A^3H_3(\omega,\omega,\omega), \end{aligned} \tag{4.19}$$

where ω is near DC.

The amplitude and phase of each output frequency component can be measured using spectrum analyzer or on-chip FFT function block. The phase information can be used to determine the sign of the transfer functions. Combining Equation 4.15 and Equation 4.19, we can compute INL from HDs analytically. If we define each harmonic distortion as the ratio of the harmonic distortion to the first order response, i.e. $HD_i = l_i/l_1$, then Equation 4.15 can be rewritten as

$$INL(x) \approx \frac{2^n - 1}{A^3}(x^2 AHD_2 + 2x^3 HD_3 - 2xA^2 HD_3 - A^3 HD_2). \tag{4.20}$$

3. Simulation-based Model Generation

We have derived closed-form models for predicting INL using simple HD measurements. However, one limitation of these models is the limited prediction accuracy especially for INL due to the weakly nonlinear assumption made and omission of other sources of nonlinearity such as quantizers. To generate more accurate predic-

tion models, we adopt a simulation-based approach where regression models predicting INL using HDs are extracted based upon a large population of simulation data. Compared to closed-form analytical models, this simulation-based approach allows a more truthful account for various circuit nonlinearities. However, the difficulty is that generating a large set of simulation data by transistor-level simulation while injecting various parametric variations is computationally infeasible. To address this challenge, we adopt the fast $\Sigma\Delta$ ADC simulation technique presented in Chapter II.

In principle, by using the simulation data, a regression model can be generated for INL based on the HD measurements. For example, we can express a specification of interest as a polynomial function of the harmonic distortions and use least square fitting to construct such model. In our case, the second and third order harmonic distortions with sign information, together with the linear system gain (to represent b_{11}) are used as the input parameters to the prediction model. We can formulate such a polynomial regression model as

$$\begin{bmatrix} 1 & H_{11} & \cdots & H_{11}HD_{21} & H_{11}HD_{31} & \cdots \\ & \vdots & & & & \\ 1 & H_{1n} & \cdots & H_{1n}HD_{2n} & H_{1n}HD_{3n} & \cdots \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} = \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} \tag{4.21}$$

where n sets of training data are used, H_{1i} is the i-th set of linear system gain (the ratio of the output signal power and the input signal power), HD_{2i} and HD_{3i} are the i-th set of second and third order harmonic distortions. The polynomial model we propose to use is of second order, so there are totally 10 terms to represent a single output. F_1 to F_n are the values corresponding to the input sets and we get the vector β from least square fitting as the coefficients for the polynomial regression model.

For the linearity test, we are mainly interested in the largest nonlinearity level in the system, so the maximum INL (INL_{max}) is a good representation for the overall

nonlinearity. The operation to find the maximum INL, however, is quite nonlinear. Therefore, a low-order polynomial regression model may not be sufficient to relate the HDs with the maximum INL. To address this issue, a more suitable regression tool, Support Vector Machine (SVM) is adopted. Support Vector Machine (SVM) [45] is a powerful method to build highly nonlinear multivariate regression models. In SVM regression, we consider a set of training data $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, where x_i is the input and y_i is the corresponding output. The input X is mapped into a high dimensional feature space using nonlinear transformation, then a best fitting function is constructed in this feature space as

$$f(x) = \omega \cdot \phi(x) + b \tag{4.22}$$

where ϕ is the nonlinear transformation, b is the bias term, and ω is the model parameter to be decided. The goal of SVM regression is to find the value of ω and b such that the values of x can be determined by minimization the regression risk.

4. Circuit Example

We demonstrate the accurate prediction of INL using simulation-based models and compare them against simpler closed-form analytical models. The test circuit is a second-order switched-capacitor $\Sigma\Delta$ ADC with 2-bit internal DAC. It is implemented in $0.13\mu m$ CMOS technology with a single 1.5 V supply. The oversampling ratio is set to 128, the sampling clock is 1MHz and the digital filters are designed to generate 11-bit digital codes. The major sources of nonlinearities considered are the internal DAC mismatch which is bounded within a typical value of 1%.

We use Equation 4.20 to calculate INL from harmonic distortions directly. Fig. 38 compares the calculated INL from HDs using the analytical expression and the what is simulated by the LUT-based simulator. We can see that the analytical model can

predict the overall trend of the INL curve very well.

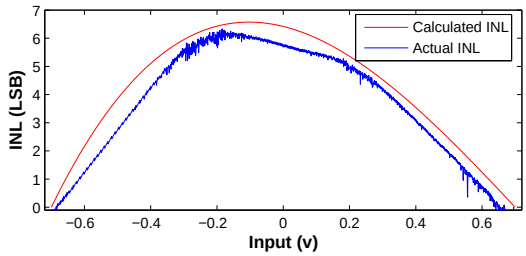

Fig. 38. Comparison of INL curves predicted by analytical model and simulated results.

The simulation-based INL prediction model is built with 1,000 simulation runs of the LUT-based simulator. In each run, the maximum INL value INL_{max} and HDs are collected. The simulation-based model is built as a SVM [45] regression model. Since it is too time consuming to use the traditional transistor-level simulation to perform a complete INL simulation, here we only compare the predicted INL_{max} values with what are simulated by our fast LUT-based simulator, as shown in Table VIII for three circuit samples.

Table VIII. The accuracy of the maximum INL prediction.

Actual INL (LSB)	Analytical (LSB)	Simulation-based (LSB)
3.5819	3.2573	3.5818
4.9884	4.7706	4.9307
3.0084	2.6986	3.0364

In Fig. 39, the accuracy of the simulation-based INL prediction model is verified for a large set of circuit instances. We can observe that the accuracy of the prediction model is very good. In this case, the average relative error is 0.69%.

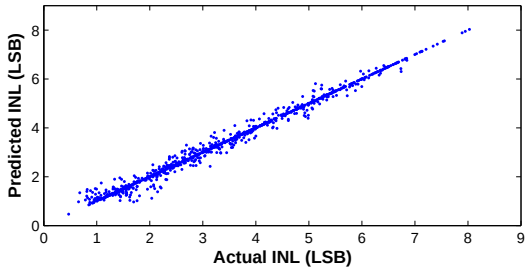

Fig. 39. The accuracy of INL_{max} prediction using the simulation-based model.

B. On-chip Test Design and Optimization for PLL

Test of phase-locked loops has been hampered by the complex mixed-signal nature of the system operation. While several on-chip test schemes have been proposed to reduce the cost of PLL test, a more systematic DFT development methodology, specially targeting at the growing parametric failures in nanometer VLSI technologies, is yet to be developed. In this section, we utilize the PLL modeling framework in Chapter II which can realistically map the device-level process variations to the variations of system-level performances. Our parametric modeling techniques allow us to examine the correlations between the system performances and specific DFT measurements feasibly through behavioral-levels simulations. An efficient methodology is developed to facilitate evaluation and optimization of PLL DFT schemes. The application of our DFT development methodology is demonstrated by generating optimized DFT schemes that produce low mis-prediction levels for detection of parametric failures of charge-pump PLLs.

1. DFT schemes for Parametric Failure Detection

The direct measurement of internal circuit nodes in a PLL is costly and it may also degrade the PLL performance [46, 47]. A better approach is to utilize the exist-

ing digital blocks, such as to use the frequency divider as counter and read out its state in order to detect chip failures [46, 47, 48, 49]. As illustrated in this chapter, the performances of different blocks are intrinsically connected. It is expected that the frequency divider/counter output will change significantly if there exists a catastrophic fault. However, parametric failures may produce smaller variations in the readout values. Hence, they are more difficult to detect and deserve more careful treatments. The three DFT schemes shown in Fig. 40 are under consideration.

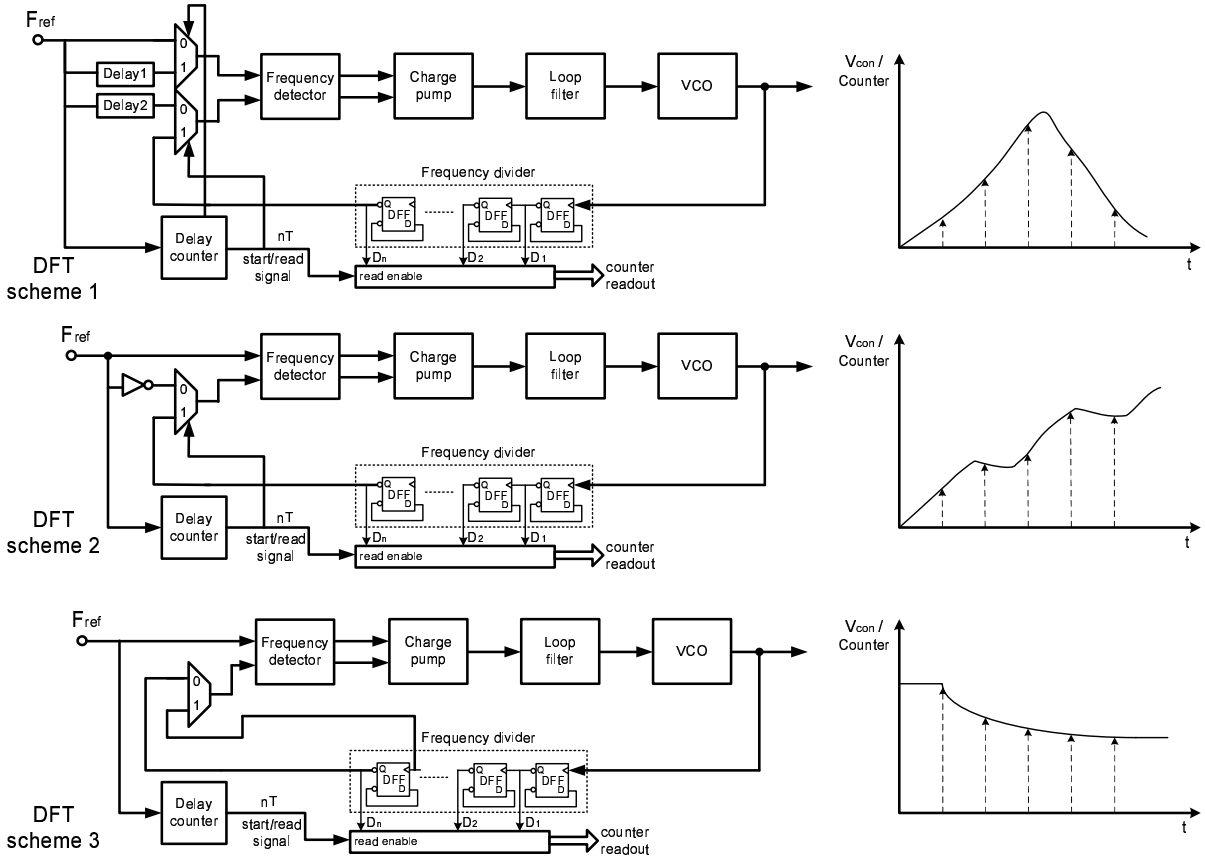

Fig. 40. DFT scheme candidates.

Similar in spirit to the existing DFT schemes, the main idea of the proposed DFT schemes is to control the charge pump in a way such that the output frequency of the PLL will be altered and the state of the frequency divider is read out at certain time

instance for failure detection. Device-level variations and mismatch will perturb the operation of the PLL and can push the system performances out of the specification window. The same parametric variations may be reflected in the variations in the readout values of the frequency divider. Parametric failures may be detected if the states of the frequency divider are strongly correlated with the design performances.

The difference of the counter output ΔN during a constant time period can be written as

$$\Delta N = \int_{T_1}^{T_2} F_{vco}(t)dt = \int_{T_1}^{T_2} f(V_{con}(t))dt \tag{4.23}$$

where T_1 and T_2 define the time interval. F_{vco} is the VCO frequency, which is a function of the control voltage V_{con} for the VCO. When it comes to the DFT scheme development, we will need to decide the way in which the the VCO control voltage is altered and define suitable test time T_1 and T_2 for failure detection. Each of the DFT schemes in Fig. 40 is discussed in details as follows.

a. Scheme 1

The first DFT scheme is similar to the one adopted in [48]. In the normal operation mode, the reference input and the output of the frequency divider are applied to the frequency detector to form the closed loop configuration. In the test mode, the output of the frequency divider is disconnected from the input of the frequency detector. The reference input and or its delayed versions are fed through the muxes to the frequency divider forming a open loop configuration. The first delay element has a larger delay value than the second one. To charge up the VCO, the reference input and its delayed version through delay 2 are applied to the frequency detector. To charge down the VCO, the delayed versions of the reference input by both delay 1 and delay 2 are selected. The delay values of the two delay elements determine the

phase error introduced at the frequency detector inputs. Hence, they also dictate the coverage of the VCO tuning range in this DFT setup. Under typical design values, delay values in the order of ten's of the reference clock signal period are required, which may cost significant silicon area to implement.

For all these three schemes, the counter read-out signals, which control the start and end points for a single test run, are generated by passing the reference clock signal F_{ref} through a series of D flip-flops. As such, the contents of the frequency divider within a defined time interval are read out.

b. Scheme 2

To solve the silicon overhead problem of the scheme 1, we propose the second DFT scheme which employs an inverter to introduce the phase difference. Since this configuration introduces a constant phase delay of π at the inputs of frequency divider, the charge pump experiences the following sequence of operation: charge up $\rightarrow$ stop $\rightarrow$ charge up $\rightarrow$ stop until the control voltage of the VCO reaches the fully voltage swing.

c. Scheme 3

The third DFT scheme is configured as follows: first the PLL is put in a standard closed-loop configuration and then a standard phase lock test is performed. Once the PLL is locked, the feedback signal frequency is changed from F_{out} to $2F_{out}$ by using the mux to select the output of the second last D flip-flop in the divider.

A brief comparison of the three DFT schemes is shown in Table IX.

Table IX. Comparison of DFT schemes.

DFT	Area cost	Test time
Scheme 1	high	short
Scheme 2	low	short
Scheme 3	low	medium

2. DFT Evaluation and Optimization

The overall flow of the evaluation and optimization of a given DFT scheme is shown in

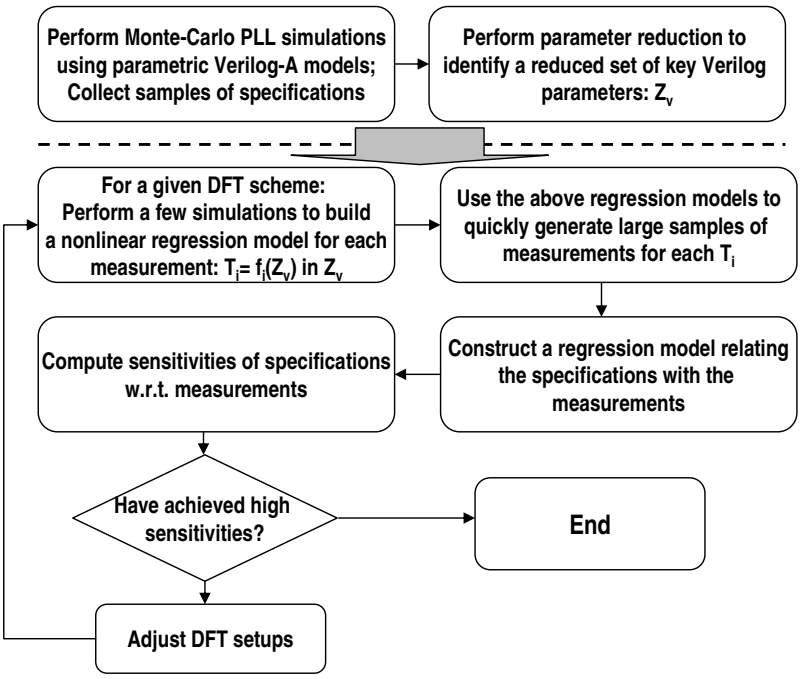

Fig. 41. Evaluation and optimization a DFT scheme.

Carlo simulations and examine the ability of using the specific DFT measurements to predict the pass/fail status of the design. Since a set of few hundred Monte-Carlo PLL simulations may take tens of hours to complete, more runtime efficient approaches are needed, especially for the optimization purpose.

a. Identification of Key System Level Variation Sources

As shown in Fig. 41, prior to the evaluation of any DFT scheme, Monte-Carlo simulations utilizing pre-characterized parametric Verilog-A models are conducted to collect a set of PLL performance samples by sampling the underlying process variations. Although being expensive, this process presents a one-time cost since it is only used to establish the correspondence between the device-level variations and variations in the system performances.

To facilitate efficient subsequent DFT evaluation and optimization steps, it is desired to identify a potentially smaller set of key system-level variation sources that contribute to most of specification variations. If such goal is achieved, the following steps can be more efficiently conducted over a compressed parameter space consisting of the most critical sources of variation. This parameter dimension task can be achieved by applying the design-specific dimension reduction technique described in Chapter II. More specifically, by utilizing the simulation data collected in the prior step, the RRR-based parameter dimension reduction is applied to identify a small set of new *critical* system-level variation sources, Z_v, which are linear or nonlinear combinations of a potentially large set of various behavioral-level model parameters of all the PLL building circuit blocks.

b. DFT Evaluation and Optimization

The optimization of a given DFT scheme with n digital outputs is illustrated in the second half of Fig. 41. Since a DFT scheme may be evaluated many times under different setups (e.g., the time interval within which the states of the frequency divider are read out) within the optimization loop, the correlation between the DFT schemes and the PLL performance must be efficiently conducted. This goal is achieved by

utilizing the critical sources of variations, Z_v, identified in the previous step.

Noticing that Z_v only contains a small set of variations, a nonlinear empirical model relating each measurement T_i of the given DFT scheme and Z_v can be rather efficiently generated. This is achieved by conducting a few Verilog-A based PLL simulations at different Z_v samples and performing nonlinear regression: $T_i = f_{tv}(Z_v)$. Note that this step does not incur a high simulation cost since regression models are only built over a low-dimension parameter space represented by Z_v. Using these easily obtained regression models, a large set of samples for each T_i can be efficiently generated. By examining the measurement data and the pre-computed system performance data, sensitivities measures can be computed to evaluate the effectiveness of the DFT scheme as follows. The overall sensitivity of the i-th system performance S_i with respect to a given DFT scheme is defined as

$$M_i = \left|\frac{\partial S_i}{\partial T_1}\right| + \left|\frac{\partial S_i}{\partial T_2}\right| + \cdots + \left|\frac{\partial S_i}{\partial T_n}\right|, \tag{4.24}$$

where n is the number of total digital outputs in the DFT, and $\frac{\partial S_i}{\partial T_j}$ is the sensitivity of S_i with respect of the j-th digital output. All these sensitivities in the above equation are obtained via least square fitting based on the measurement and performance data mentioned before. Note that $\frac{\partial S_i}{\partial T_j}$ is normalized with respect to the variance of T_j. For a given DFT scheme, efficient optimization is conducted to find an optimal setup that leads to a highest overall sensitivity.

Once an optimal DFT scheme has been identified, a more accurate correlation model that is capable of accurate pass/fail chip prediction using the corresponding DFT measurements is extracted. To capture the potential nonlinear correspondence between the design performances and the measurements, Support Vector Machine (SVM) is adopted as an accurate classifier. Support Vector Machine (SVM) [45] is a powerful method to build highly nonlinear multivariate regression/classification mod-

els. In SVM regression, we consider a set of training data $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, where x_i is the input vector and y_i is the corresponding output. The input X is mapped into a high dimensional feature space using nonlinear transformation, then a best fitting function is constructed in this feature space as

$$y = f(x) = \omega \cdot \phi(x) + b \tag{4.25}$$

where ϕ is the nonlinear transformation, b is the bias term, and ω represents the model parameters to be decided. Based on this nonlinear function $f(\cdot)$, we can classify the chip as faulty or not with the DFT circuit outputs.

3. Optimization Example

We demonstrate the application of the proposed DFT development methodology with a PLL design example. The PLL circuit and proposed on-chip test schemes are implemented in 90-nm CMOS technology, specifications of the PLL are listed in Table X.

Table X. PLL specifications.

Technology	90 nm CMOS
Supply Voltage	1.2 V
Power Consumption	0.98 mW
Reference Signal	10.7 MHz
Center Frequency	1.37 GHz
Frequency Divider Ratio	128

We consider three system performances. The first one is the startup time from power on to the locked mode, which is actually the lock time from the zero frequency condition to the operation frequency. We denote it as $locktime1$. The second spec-

ification $locktime2$ is mainly targeted for the dynamic behavior of the PLL, which is measured as the time needed from the locked mode to the new locked mode with half output frequency. The third specification considered is the maximum frequency $maxfreq$, which determines the highest clock frequency that the PLL can generate. The device-level parameter variations with spatial correlations are modeled following the work in [50, 51]. We use multivariate Gaussian distributions to model various transistor parameter variations such as the variations of the threshold voltages. For each variational parameter, the variance is set to $3\sigma = 10\%$ for 90-nm CMOS technology [27].

a. Performance Modeling

The frequency versus control voltage curve of the VCO is extracted using the efficient modeling framework in Chapter II. First we simulate the VCO for a few clock cycles and gather the time-domain output response as Y. Then RRR is applied to get a reduced parameter set Z to represent the important device-level parameters. A parametric model of the VCO in terms of Z is then built. To model the statistical characteristics of the VCO accurately, a 6-th order polynomial fitting is used to fit the output frequency vs. control voltage curve.

The Verilog-A models for other building blocks are extracted in a similar fashion. Specifically, the charge pump model is generated using a 3-rd order polynomial in the output voltage for each charge-up/down current. There are a total of 17 Verilog-A model parameters extracted for the complete PLL design.

b. Test Scheme Evaluation and Optimization

Prior to the DFT evaluation and optimization, we perform Monte-Carlo simulations to collect a large set of system performances samples. This is accomplished using our

efficient parametric modeling infrastructure. The distribution of each specification is shown in Fig. 42. The pass/fail decision is based on whether all specifications can meet the performance windows or not. The targeted specification windows are set as follows: $locktime1 < 2.5\mu s$, $locktime2 < 2.5\mu s$ and $maxfreq > 2.085GHz$.

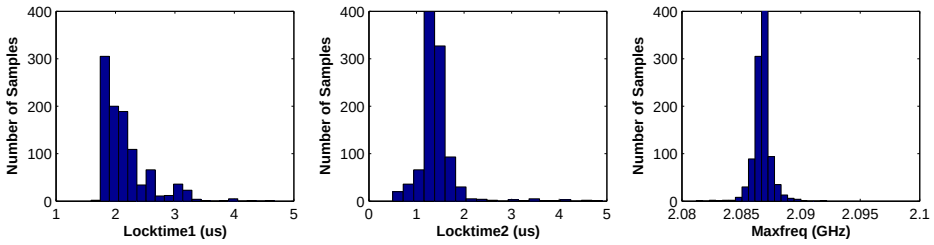

Fig. 42. Distribution of system performances.

As discussed previously, each DFT scheme is based on reading out the states of the frequency divider and the corresponding digital codes are used to identify parametric failures. Since the bit length of each digital code is limited by the number of D flip-flops in the frequency divider, overflow may occur. When overflow happens, we add 2^m to compensate it, where m is the number of D flip-flops in the frequency divider. The proposed DFT evaluation and optimization methodology is applied to each of three DFT schemes. At the end of the optimization, we rank the effectiveness of each test scheme by computing the total DFT sensitivity to each specification, as listed in Table XI. Our analysis results indicate the scheme 1 is the most effective

Table XI. Sensitivities of system performance to DFT schemes.

	Spec. 1	Spec. 2	Spec. 3
DFT Scheme 1	4.7439	2.457	1.8836
DFT Scheme 2	2.3752	0.6124	0.0905
DFT Scheme 3	2.6324	1.4951	1.2698

scheme since it has the largest sensitivities for all the specifications. Scheme 2 is identified as the least effective scheme.

c. DFT Scheme Verification

The three DFT schemes have been ranked in the previous section using sensitivity analysis. To verify this result, for each scheme, a SVM model is extracted to predict the pass/fail status of the chips based on the corresponding DFT outputs. 400 Monte-Carlo simulation samples are generated by conducting PLL system simulation using Verilog-A macromodels. These data are used to generate the SVM model. To evaluate the effectiveness of the each scheme more reliably, another 100 Monte-Carlo simulations are carried out and used as the test data for checking the accuracy of the SVM model. The pass/fail predictions achieved through the three SVM models are compared against the simulated chip performances, as shown in Fig. 43. Here, the predictions made through the simulation are labeled as "direct measurement", and $+1$ indicates a chip being classified as "fail" while -1 indicates the opposite. Some offset has been applied to each DFT scheme when fail chip detected (at points of $+1$) for easier comparison.

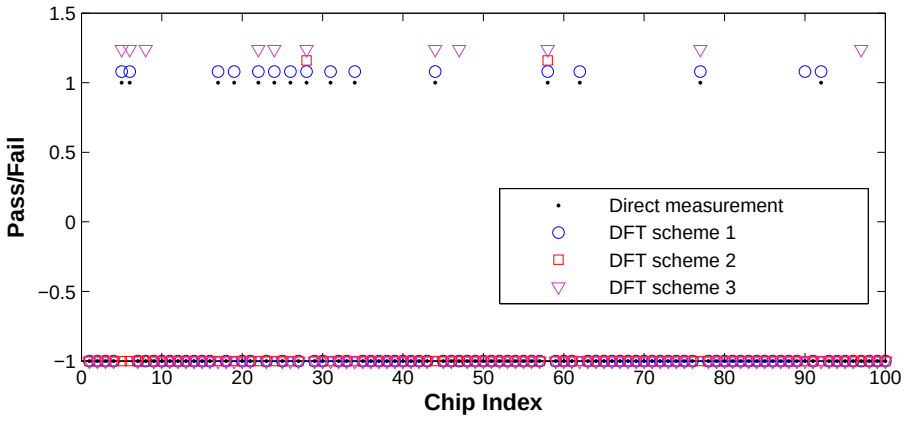

Fig. 43. Pass/fail predictions of three DFT schemes.

From Fig. 43 we can see that DFT scheme 1 only has only 1 misclassification. The performance of DFT scheme 2 is verified to be poor as it can only detect two faulty chips which may have largest variations. DFT scheme 3 can detect more failures than scheme 2 but is still not as good as scheme 1. The performance of each DFT scheme are summarized in Table XII. The Monte-Carlo simulation results confirm the validity of our sensitivity analysis.

Table XII. Comparison of DFT schemes to identify faulty chips.

	Defect escape	Yield loss	Overall accuracy
DFT Scheme 1	0.0%	1.0%	99%
DFT Scheme 2	13.0%	0.0%	87%
DFT Scheme 3	7.0%	3.0%	90%

The faulty chips that can not be detected by various DFT schemes are mostly near the system specification boundaries. This can confirmed by examining the locations of mis-predictions in the specification space. Such an analysis is done for schemes 1 and 3 in Fig.44 and Fig.45, where the dashed cube represents the acceptance region of the specifications.

For an effective DFT scheme, the change of process parameters shall be reflected in the digital outputs such that the potential parametric failures may be detected. To examine this, we consider two specific perturbed PLL circuits. In the first circuit, relatively small parametric variations are introduced in the VCO and the circuit is verified to be meeting all the specifications. While in the second circuit, large perturbations are introduced into the charge-pump, causing parametric failures. We show the digital outputs produced by DFT scheme 1 for the nominal circuit and the two perturbed circuits in Fig. 46. As can be seen, the digital outputs of the first

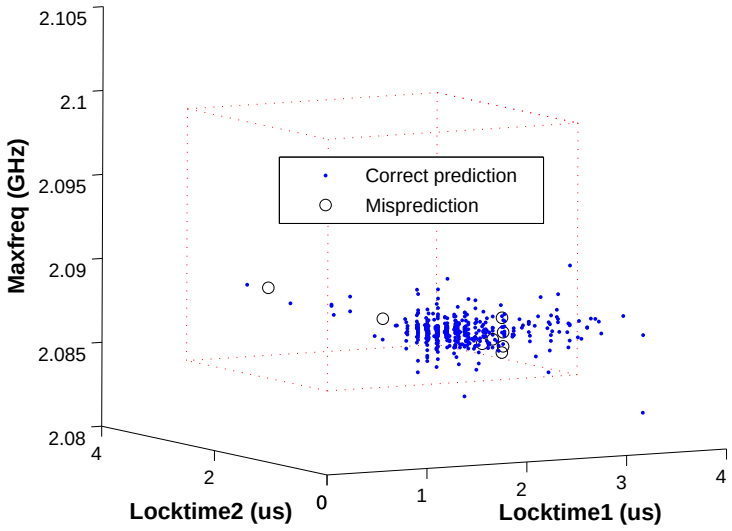

Fig. 44. Chip prediction distribution for DFT scheme 1.

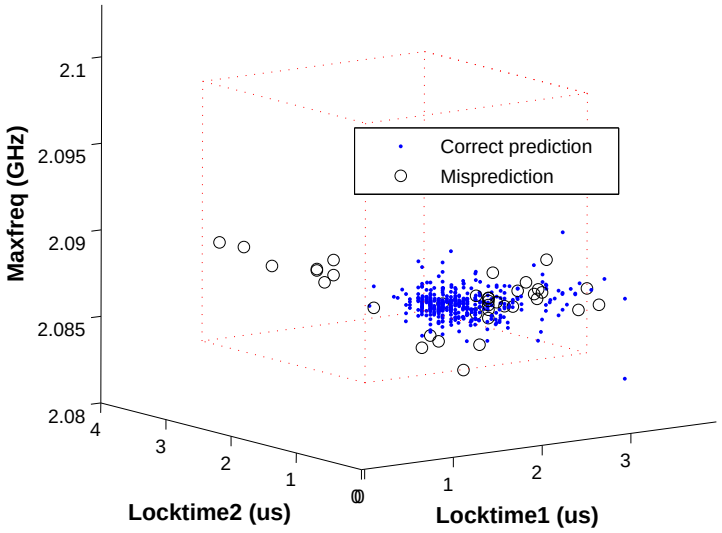

Fig. 45. Chip prediction distribution for DFT scheme 3.

circuit do not vary significantly from those of the nominal circuit and the circuit is classified correctly as a good chip by the DFT results. In contrast, the digital outputs of the second circuit vary significantly from the nominal values, which is also correctly classified by the DFT scheme as a failing chip.

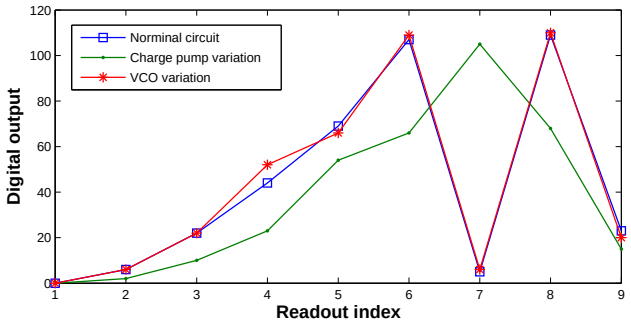

Fig. 46. Digital output changes due to process variation for DFT scheme 1.

d. DFT Trade-off Analysis

We further look into the trade-off between the accuracy and the number of digital outputs for the considered DFT schemes. This is important since fewer test codes will correspond to a shorter test time, if a similar accuracy can be achieved. This trade-off analysis is conducted for every scheme in Fig. 47. It can be observed that for a small number of digital outputs, the accuracy of scheme 2 is actually higher than that of scheme 3. It can be also seen that the accuracy of scheme 3 becomes quickly saturated as the number of outputs increases. Under all the cases, scheme 1 is always the optimal choice.

C. Summary

In this chapter we address the problems of testing analog/mixed-signal circuit performances with consideration of process variations. We propose to use Volterra series

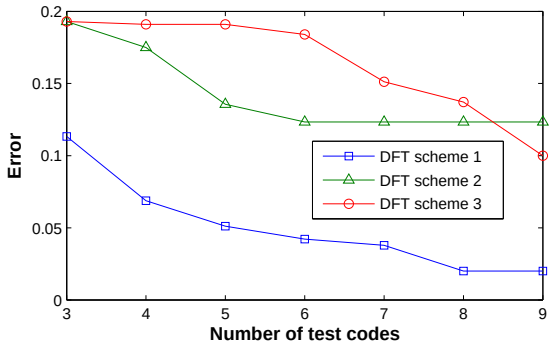

Fig. 47. Error v.s. number of test codes.

to analyze the nonlinear circuit behaviors of Sigma-Delta ADCs in the frequency domain. This analysis allows us to formally relate the linearity metrics with easily obtained harmonic distortions measurements. The closed-form analytical models have been derived to allow lost-cost linearity test based on simple HD measurements. In order to further improve the prediction accuracy, a simulation-based model generation approach is adopted which is enabled by the fast LUT-based ADC simulation technique presented in Chapter II. The good accuracy of the proposed prediction models is verified by the close fitting of the INLs obtained by the proposed linearity test method and the actually simulated values.

Design-for-test development methodology targeting at the detection of parametric failures in charge-pump PLLs is presented. Such methodology is enabled by detailed bottom-up macromodeling which leads to a scalable parametric PLL simulation infrastructure based on Verilog-A behavioral modeling presented in Chapter II. In conjunction with the powerful dimension reduction techniques that are employed to deal with high process variation space, efficient PLL DFT scheme evaluation and optimization are conducted. As a demonstration of the proposed techniques, three DFT schemes are evaluated and optimized. Monte-Carlo simulations are performed, which confirm the results obtained from our DFT methodology. Furthermore, trade-

off analysis concerning detection accuracy and test time for the DFT schemes are also performed.

CHAPTER V

DESIGN CASE: ALL-DIGITAL PLL

In this chapter we apply the modeling, optimization and testing techniques discussed in the previous chapters in a complicated all-digital PLL (ADPLL) design case. We start with understanding the operation of ADPLL system, then implement the building blocks in transistor level. System optimization considering the uniqueness of digital intensive implementation and system reconfigurations is performed with efficient variation-aware block models. On-chip performance detection and performance tuning functions are also implemented to detect parametric failures and carry out performance compensation.

A. System Background

The concept of ADPLL was first proposed for clock generation with integer-N frequency multiplication [52, 53]. With the digital interface, PLL frequencies can be controlled directly by digital logic blocks or microcontrollers. Since the clock generation does not require very accurate frequency steps, most of these systems featured ring oscillators with integer-N frequency multiplication. The recent proposed ADPLL designs are mainly targeted at wireless communication applications, which have tight system specifications for jitter, power and frequency resolution [54, 55, 56, 57]. Most of these systems feature fractional-N frequency multiplication and employ LC-tank oscillators to achieve good phase noise performances. In this chapter, we focus on the design of fractional-N ADPLLs with LC-type oscillator cores.

An example of ADPLL block diagram is shown in Fig. 48. The system output frequency F_{out} is determined by the frequency division ratio (FDR). Suppose N_i is the integer part of the frequency division ratio and N_f is the fractional part, the system

output frequency F_{out} equals to $FDR = N_i + N_f$ times the reference signal frequency F_{ref}.

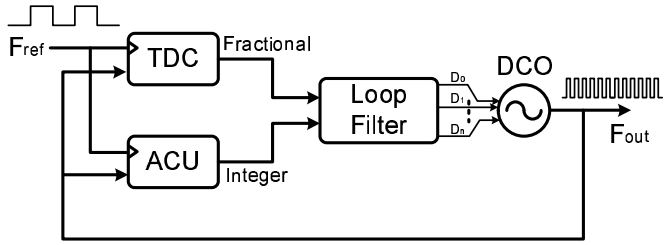

Fig. 48. All-digital PLL system block diagram.

The core of ADPLL is digital controlled oscillator (DCO), of which the frequency is controlled by digital codes. The control words for DCO come from the phase difference of reference signal and the output signal. During a reference clock period, clock accumulator (ACU) is used to count the integer number of the output clock F_{out} cycles in this reference clock period, and time-to-digital converter (TDC) is used to calculate the fractional part.

If PLL is stabilized, the accumulated clock cycle number should be equal to the frequency division ratio.

$$ACU[k] - ACU[k-1] + TDC_{gain} \cdot (TDC[k] - TDC[k-1]) = N_i + N_f \tag{5.1}$$

where $ACU[k]$ and $TDC[k]$ are ACU and TDC outputs in the k-th clock cycle, TDC_{gain} is the scaling factor to convert TDC outputs from digital measurement results to the frequency domain. Suppose the resolution of TDC is T_{res} and the output clock period is T_{out}, TDC gain can be calculated as [58]

$$TDC_{gain} = \frac{T_{res}}{T_{out}} = T_{res} \cdot F_{out} \tag{5.2}$$

If the counted output clock cycles do not meet the required frequency division ratio, the difference detected by ACU and TDC will be passed through loop filter to

adjust DCO frequency. The digital filter in the loop is to make the frequency tuning smooth. In ADPLL systems, the digital implementation of loop filters can eliminate large capacitors needed in conventional charge-pump PLL and is robust to process variations.

In real world, there always exist nonlinearities and noises in building blocks like thermal noise in DCO and limited time resolution in TDC. These nonlinearities are noise-shaped by loop filters at system output as shown in Fig. 49. Careful design of loop characterizations can reduce the total phase noise by balancing the contributions from different noise sources, so the filter design is very critical for the ADPLL performance.

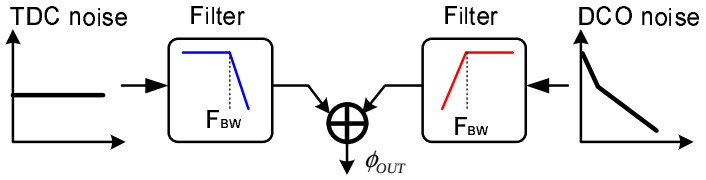

Fig. 49. Phase noise contributions of TDC and DCO.

ADPLL system design is mainly focused on constructing key building blocks. There are different topologies for each building block and many trade-offs to consider. For example, we can use a high-resolution TDC for better frequency detection with more power burned, but whether it is a good choice really depends on system specifications. The trade-off analysis becomes more complicated when process variations are taken into consideration. The varying transistor characterizations make system performances statistical variables, so the designs tuned in nominal operation may not be optimal under process variations. To solve these problems, we first explore possible architectures of building blocks and evaluate their system performances. Then we perform yield-aware transistor tuning to optimize circuit design under specified system configurations.

B. System-level ADPLL Design

Traditionally, PLL designers would choose a topology for PLL based on the open loop transfer function then set the gain and pole/zero locations to achieve the required phase/gain margin [59]. The choice of building blocks and loop filter characterizations is the first step in ADPLL design. Since the number of possible choices is limited, it is beneficial to evaluate the system performances with these discretized topology parameters as the starting point.

1. System Performance Analysis

It is very difficult or impossible to use SPICE-like simulators to evaluate ADPLL system performances due to the extraordinarily long computation time, which may take weeks or months for a single transient run. Transfer function based simulation approach is proven to be accurate and efficient to evaluate phase noise [7, 59, 60] if block noises are modeled accurately.

Since ADPLL operates in the digital domain, the corresponding transfer function is z-operator. A Backward-Euler transformation from z-domain to s-domain can be written as in Eqn. 5.3 for small ω [7]. This approximation is valid as long as the frequencies of interest are much smaller than the sampling rate, which is F_{ref} in this case. It is widely accepted that this linear approximation holds as long as the PLL bandwidth F_{BW} is at least 10 times smaller than the sampling rate [61]. Here the PLL bandwidth is set to a few ten KHz while the reference signal frequency is over ten MHz, so the transformation in Eqn. 5.3 holds the accuracy.

$$z = e^{j\theta} \approx 1 + j\theta = 1 + \frac{j\omega}{F_{ref}} = 1 + \frac{s}{F_{ref}} \tag{5.3}$$

The noises of building blocks can be modeled and injected to ADPLL systems

as in Fig. 50 [7]. Noises will go through high-pass or low-pass shaping depending on the different locations they are injected in.

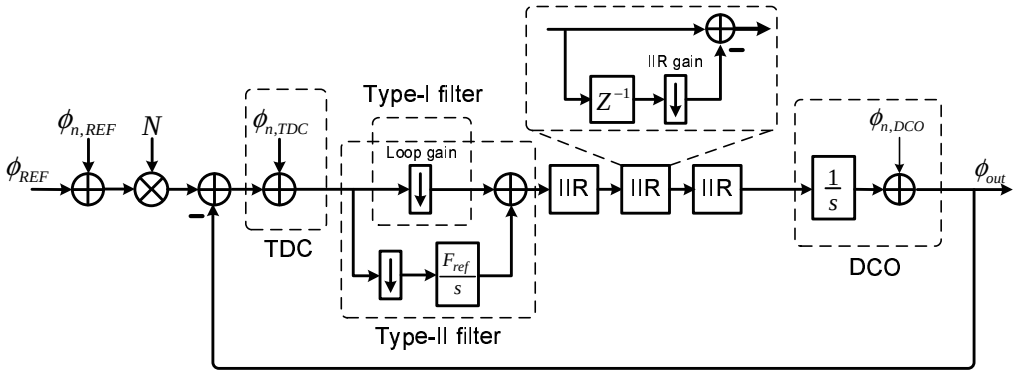

Fig. 50. s-domain linear ADPLL noise model.

Suppose we have loop filter transfer function $H_{LF}(f)$, the TDC noise at the system output $N_{o,TDC}$ can be written as

$$N_{o,TDC} = \left| \frac{H_{LF}(f)}{1 + H_{LF}(f)} \right|^2 \phi_{n,TDC} \tag{5.4}$$

where $\phi_{n,TDC}$ is the total TDC noise. The DCO noise at the system output $N_{o,DCO}$ can be written as a function of total DCO noise $\phi_{n,DCO}$

$$N_{o,DCO} = \left| \frac{1}{1 + H_{LF}(f)} \right|^2 \phi_{n,DCO} \tag{5.5}$$

Similarly the reference signal noise contribution $N_{o,REF}$ is

$$N_{o,REF} = \left| \frac{N \cdot H_{LF}(f)}{1 + H_{LF}(f)} \right|^2 \phi_{n,REF} \tag{5.6}$$

where N is the frequency division ratio and $\phi_{n,REF}$ is the reference signal noise.

Total system phase noise $N_{o,tot}$ can be calculated by adding these noise contributions together

$$N_{o,tot} = N_{o,TDC} + N_{o,DCO} + N_{o,REF} \tag{5.7}$$

As can be seen in Eqn. 5.7, loop characterizations and block noises determine

system noise performances. Other system performances like power and area are much easier to be included in the system level, as long as we know the corresponding numbers of building blocks, they can be simply added in the system level. So in the following sections we will put our focus on noise performance modeling.

2. Loop Filter

The filters in ADPLL systems can be configured as Type-I or Type-II, which have different phase noise shaping capabilities. The transfer function of type-I filter can be considered as a direct feedforward path with a tunable loop gain α. Type-II filter has an extra pole and its transfer function can be written as [7]

$$H_{LPF}(f) = \alpha + \frac{\rho \cdot F_{REF}}{s} \tag{5.8}$$

where ρ is the filter coefficient.

High order filters can also be added for better phase noise suppression. IIR filters are typically used for the simplicity [7]. To ensure loop stability, cascading of single-pole IIR filters can be used. For a M-th order IIR filter, the transfer function can be expressed as

$$H_{IIR}(f) = \left(\frac{1 + s/F_{REF}}{1 + s/\left(\lambda F_{REF}\right)} \right)^M \tag{5.9}$$

where λ is the IIR filter gain.

Digital filters are superior to traditional analog implementations since they are robust to process variations, so we can save the effort of modeling of process variations for loop filters. For loop coefficients/gains, numbers in the powers of 2 are much easier to be implemented in digital circuits, so we select these discretized values in the system design space exploration.

3. Time-to-Digital Converter

Time-to-digital converters in ADPLL behave like phase detectors in charge-pump PLLs to detect the frequency/phase differences between input signals. The basic principle of TDC is that the fast input signals are passed through delay chains and the comparators are used to detect the point when the two signals exchange leading. The nonlinearities of TDC are mainly due to the limited time resolution, which equals to the minimum cell delay T_{res}. It behaves quite like the quantization noise in analog-to-digital converters (ADCs). For an ideal TDC, the noise power can be written as [58]

$$\phi_{n,TDC} = \frac{1}{12 \cdot F_{REF}} \left(\frac{2\pi \cdot T_{res}}{T_{out}} \right)^2 \tag{5.10}$$

As can be seen from Eqn. 5.10, TDC noise power is proportional to the square of TDC time resolution T_{res}, which is typically a buffer or inverter delay. So it is of interest to use higher resolution TDC implementation to reduce associated noise with a penalty of extra power [57, 62]. The choice of fine resolution TDC is also treated as a topology selection variable in the ADPLL system design exploration.

Process variations will cause nonlinear time-to-digital conversion, which makes the TDC noise level increase and causes additional noise in the ADPLL system output. We will discuss this effect and introduce a numerical method to calculate the influence in the next section.

4. Digital Controlled Oscillator

The nature of DCO is still oscillator, so the flicker noise and thermal noise inevitably affect DCO performances. These natural oscillator noises can be characterized using simulators like SpectreRF [63] and their influences at the system output can be calculated using Eqn. 5.5.

As the frequency control of oscillator is implemented by the capacitance tuning, the minimum varactor capacitance switching determines the frequency resolution. Due to the size of varactors, the minimum frequency resolution achievable by switching varactors is a few ten KHz [7]. Better frequency resolutions can be achieved by using digital-to-analog converters (DACs) to control the unit varactor capacitance continuously [56] or applying Sigma-Delta modulation (SDM) for varactor banks [7]. For the DAC based approach, the extra noise power can be written as [60]

$$\phi_{n,DAC}(f) = 4KTR_{eq}\frac{1}{1+(f/f_p)^2} \tag{5.11}$$

where R_{eq} and f_p are the equivalent resistance and corner frequency of the DAC. The noise power of the SDM approach is [7]

$$\phi_{n,SDM}(f) = \frac{1}{12}\left(\frac{f_{res}}{f}\right)^2 \frac{1}{f_{dth}}\left(sinc\frac{f}{f_{dth}}\right)^2 \tag{5.12}$$

where f_{res} is the DCO frequency resolution and f_{dth} is the dithering frequency. The selection of DAC or SDM approach is used as a discretized system design parameter.

C. Block Modeling in ADPLL

In this section, we model the performances of TDC and DCO with different design variables and process variations. The models presented in this section are used in both transistor-level fine tuning and system-level performance distribution analysis.

1. TDC Modeling

Process variations will cause mismatch between TDC cells, which in turn makes the time-to-digital transfer curve nonlinear. The process variation induced nonlinear transfer curve will cause extra phase noise in addition to the inherent quantization

noise in Eqn. 5.10 [64], so careful analysis is needed for the TDC modeling under process variations.

From Eqn. 5.1 we can see that TDC output $TDC[k]$ travels through all the digital codes with equal probabilities as long as $N_f \neq 0$. The accurate time steps in TDC transfer curve can be obtained using the transistor-level simulation. With a determined TDC cell delay distribution, we can calculate the TDC noise numerically. As shown in Fig. 51, we calculate the power of the statistical variable which deviates from the ideal transfer curve, and add its power along the whole range of input time differences to get the total TDC noise.

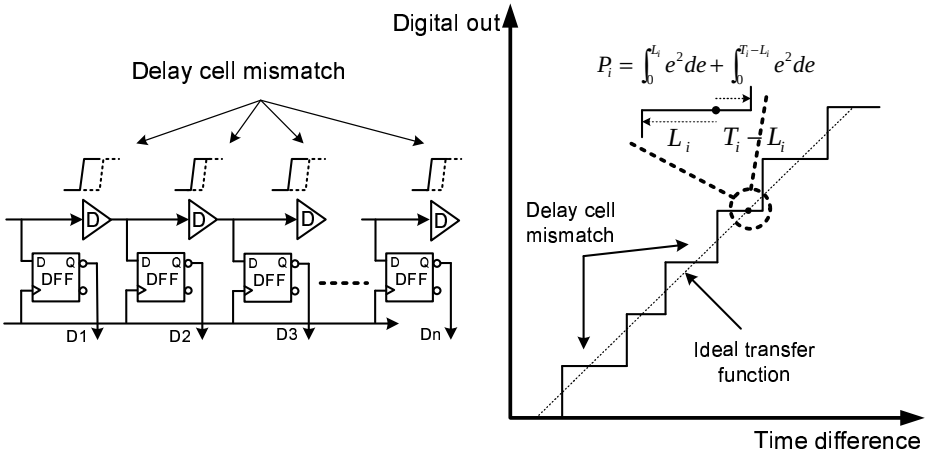

Fig. 51. Modeling of TDC noise.

Detailed analysis and calculation of TDC power are presented as follows. For the i-th TDC step, we start from the interception point of the ideal transfer curve and the real transfer curve and set the two ends as L_i and $T_i - L_i$. With the assumption that the input time is evenly distributed, the noise within this time step equals to the probability of e deviates from the ideal point

$$P_i = \int_0^{L_i} e^2 de + \int_0^{T_i - L_i} e^2 de \tag{5.13}$$

The total noise of the TDC can be calculated by adding noise in every TDC delay cell and averaging it over the whole TDC input range which equals to the output signal period

$$\phi_{n,TDC} = \frac{1}{T_{out}} \sum_{i=1}^{M} P_i \tag{5.14}$$

here M is the total number of TDC cells.

There are a few constrains to be noticed when using Eqn. 5.13 and 5.14. First, L_i and T_i are related geometrically, Eqn. 5.15 shows the constraint when the ideal transfer curve has a unity slope. Second, the ideal transfer curve is "best-fitting curve" the coefficients of which can be calculated by minimizing the total noise in Eqn. 5.14. Third, since the total TDC cell number $M = T_{out}/T_{avg}$ may not be an integer number, the last digit of TDC will contribute higher noise. Here T_{norm} is the averaging single TDC cell delay.

$$(T_{i-1} - L_{i-1}) + L_i = T_{avg} \tag{5.15}$$

As discussed in the previous section, using fine resolution TDC can cut down the TDC noise. Another option for noise reduction is to search for sets of transistor designs which can reduce TDC cell mismatches under process variations. So we model TDC performances with design and process variables targeting for statistical optimization. A formal TDC model is defined as

$$\overrightarrow{P}_{TDC}(\overrightarrow{T}_{TDC}, \overrightarrow{D}_{TDC}, \overrightarrow{V}) \tag{5.16}$$

where $\overrightarrow{P}_{TDC}$ represent TDC performances, $\overrightarrow{T}_{TDC}$ account for the TDC topology selection, $\overrightarrow{D}_{TDC}$ are transistor sizes in the TDC design and $\overrightarrow{V}$ stand for process variations.

2. DCO Modeling

Inherent DCO noise contributes a significant part of total system noise. Generally phase noise/jitter in DCO drop as the current through LC-tank increases (SNR increases with higher signal power), but the phase noise will go up again when the current is too large and output signals start clipping. The modeling of DCO phase noise is illustrated in Fig. 52.

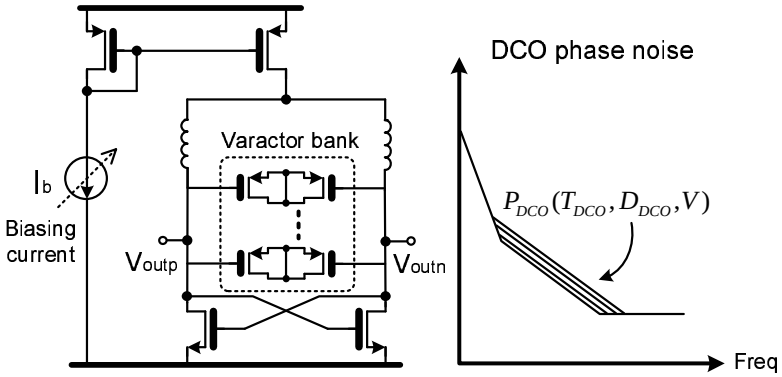

Fig. 52. Modeling of DCO noise.

Larger biasing current results in higher power, this trade-off needs to be considered in the ADPLL optimization. So we model the DCO phase noise and power in terms of biasing current and transistor sizes. Process variations also have an impact on DCO noise performances and should be included in the performance model. We formulate the DCO model as

$$\overrightarrow{P}_{DCO}(\overrightarrow{T}_{DCO}, \overrightarrow{D}_{DCO}, \overrightarrow{V}) \tag{5.17}$$

where $\overrightarrow{P}_{DCO}$ represent DCO performances, $\overrightarrow{T}_{DCO}$ account for the choice of SDM or DAC to increase the ADPLL frequency resolution, D_{DCO} represent the transistor sizes and biasing current and V stand for process variations

D. Yield-aware ADPLL Optimization

With ADPLL building block models and transfer functions obtained in the previous sections, we introduce the yield-aware optimization framework for optimal ADPLL design. Costly system performance analysis are replaced by the efficient simulation framework using s-domain circuit models. We first evaluate the performances of different topologies for each building block, then perform the refined device-level variable tuning scheme to achieve the optimal system performances.

1. Topology Selection

Since there are different choices of circuit blocks to build an ADPLL system, it is of benefit to choose the suitable topology combinations for ADPLL systems first. These topology selection variables are discretized numbers $\overrightarrow{T} = \{T_{DCO}, T_{TDC}, T_{Filter}\}$. The primary goal in topology evaluation stage is to find a reduced set of possible topologies for faster localized design parameter tuning, and generate a picture of system performance trade-offs.

There are always trade-offs in circuit designs. It is almost impossible to achieve a solution of best performance for each specification. To solve this problem, we construct multi-objective system cost functions to balance different performance requirements. We explore system performances in the discretized topology space by evaluating the cost functions and then achieve the optimal solutions. We do not stop at the design combination with the lowest cost function as the only optimum solution. This is because some "near-optimal" discretized design points may have better statistical performances when we perform block tuning and take into the consideration of process variations. So a few discretized system design choices are kept, and fine tuning of building blocks are carried out for these points. Retaining a few

tens of discretized design choices in the fine tuning stage would be good enough for optimization in practice.

2. Yield-aware Fine Tuning

After the topology selection, we have a few optimized discretized system topologies to choose from, denoted as $\overrightarrow{T}_{opt}$. In this stage we perform the tuning of TDC and DCO within the reduced discretized system space.

Due to the process variations, system performances also become statistical variables. For k-th statistical system performance P_k (smaller the better), suppose we need a yield of Y_k, then the yield-aware performance $P_k^{Y_k}$ satisfies the following probability condition

$$\mathrm{P}\{P_k \leq P_k^{Y_k}\} = Y_k \tag{5.18}$$

Eqn. 5.18 implies that for P_k, the best achievable performance value is $P_k^{Y_k}$ when yield level Y_k is required. $P_k^{Y_k}$ is considered as the yield-aware k-th system performance and to be used in the overall system optimization.

Since we perform fine tuning for each discretized system configuration set, the yield-aware fine tuning can be formulated as minimizing the cost function in the building-block space within

$$\min Cost\left(\overrightarrow{P}_{TDC}(\overrightarrow{D}_{TDC}), \overrightarrow{P}_{DCO}(\overrightarrow{D}_{DCO})\right)|_{\overrightarrow{T}_{opt}} \tag{5.19}$$

where design variables of TDC and DCO models are selected as input variables to minimize the statistical system cost function. Any suitable optimization packages can be used for this minimization problem. In this chapter, we employ the multiple coordinate search algorithm in [34]. The proposed two-step optimization flow is summarized and illustrated in Fig. 53.

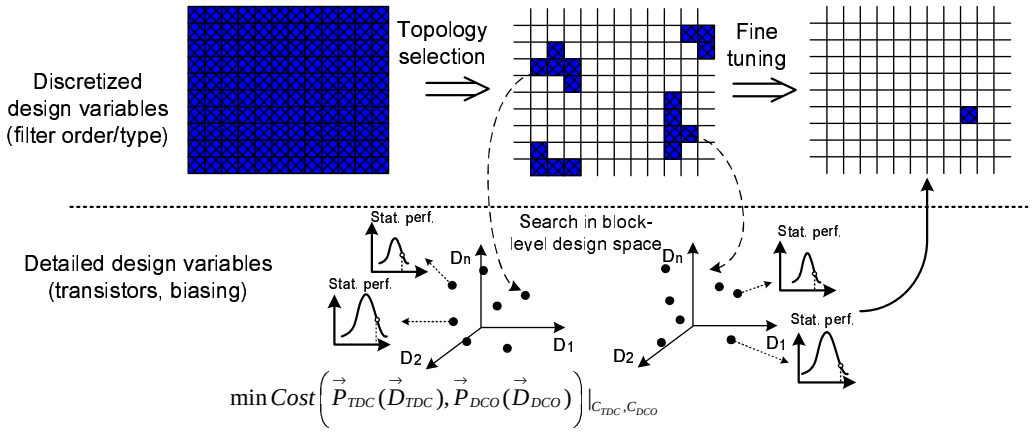

Fig. 53. Yield-aware optimization flow for ADPLL.

E. Adaptive Self-tuning ADPLL Design

System performance monitoring needs to be addressed before any compensation block can be designed. Thanks to the digital implementation, all the control signals in ADPLL systems are in the digital domain and easy to be processed. When the ADPLL loops are stabilized, the frequency errors measured by the TDCs are found to possess of strong correlation to the system phase noises [65]. The TDC itself behaves like a jitter measurement block since it detects the differences between the reference signal edges and the output signal edges. So we use the RMS value of frequency differences (FE_{RMS}) as the indicator of system jitter performances. For simplicity, the RMS value is calculated for every 64 clock cycles.

The goal of using FE_{RMS} as system jitter indicator is to facilitate cheap performance monitoring. A predefined threshold number for frequency error (FE_{TRE}) is calculated in the design stage by statistical simulation and can be further calibrated using silicon measurement results. If the frequency error is large than the threshold value, it indicates that the system needs to be reconfigured to enhance the performances. As mentioned earlier, the power consumption for the fine resolution TDC

is pretty high, so in the system level design we need to perform trade-off analysis to see whether the fine resolution TDC should be turned on or off. Loop configurations, including loop gain and loop filter orders can be modified without power penalty, so they are free of tunability. The adaptive PLL system diagram is shown in Fig. 54. The DCO biasing current is calibrated off-chip for the current design.

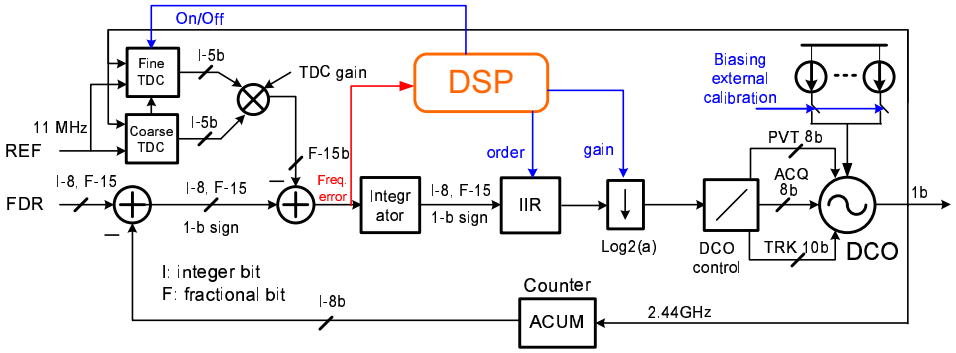

Fig. 54. Adaptive PLL system diagram.

The detailed logic control sequence of the performance self-compensation function is shown in Fig. 55. When the self-healing block detects $FE_{RMS} > FE_{TRE}$, it first reconfigures loop gain within $[2^{-10}, 2^{-9}, 2^{-8}, 2^{-7}]$ and loop order within $[1, 2]$, a total of 8 different combinations. For each new configuration, we wait for T_1 cycles to let loop gets stabilized and calculate FE_{RMS} in the next T_2 cycles. If the minimum FE_{RMS} in these 8 configurations is lower than FE_{TRE}, the compensation algorithm will stop and save the configuration with the lowest FE_{RMS} as the new loop configuration. If the best FE_{RMS} still higher than FE_{TRE}, fine-resolution TDC will be powered on and the system will go through the 8 loop configurations. The configuration with the best FE_{RMS} will be saved to reconfigure the ADPLL system.

The performance monitoring and control function blocks are designed in Verilog HDL and synthesized using ARM Artisan standard cell library. Total gate counts for the block implementation is about 10,000.

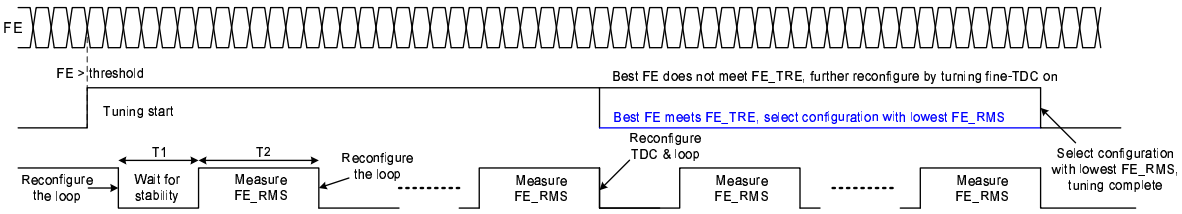

Fig. 55. Logic sequence of self compensation.

By employing the self-compensation operation in ADPLL systems, we can detect the failure chips and make compensation to bring them back to meet the requirements. Employing this approach has the potential to achieve better overall system performances than the synthesis of analog circuits by only sizing transistor sizes. Conventional analog optimizations for yield enhancement tend to push designs to high performance corners. This approach works since the chips with performance dropping induced by process variations can be tuned to meet the specifications. However, as discussed in [32] and [24], these improvements are often achieved with the sacrifice of other performances. Fig. 56 illustrates one of the scenarios where power increases and the overall yield may even drop if power specification is tight.

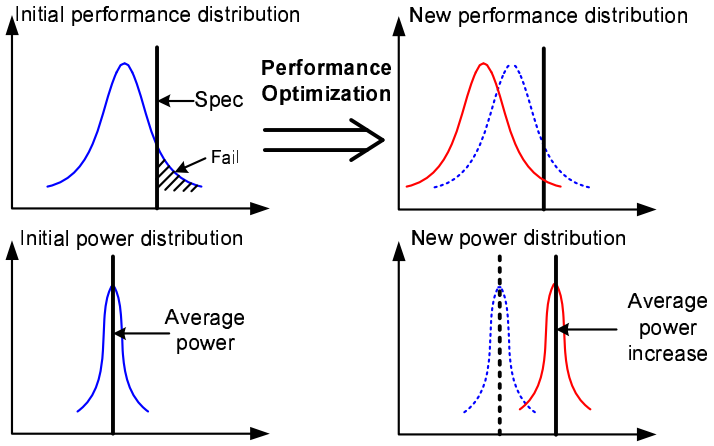

Fig. 56. Conventional yield-aware optimization.

The self-healing function in the adaptive ADPLL systems works by switching to higher performance configurations only when process variations cause the chips fail

to meet the specifications, so we do not need to design the system in the high performance and high power configurations to fight for process variations. As illustrated in Fig. 57, adaptation kicks in when the chips fail to meet the required performance targets, which is statistically rare. The new circuit configuration burns more power only for the portion of chips not meeting the specifications, so the statistical power consumption can be improved when compared with conventional yield-aware optimization results.

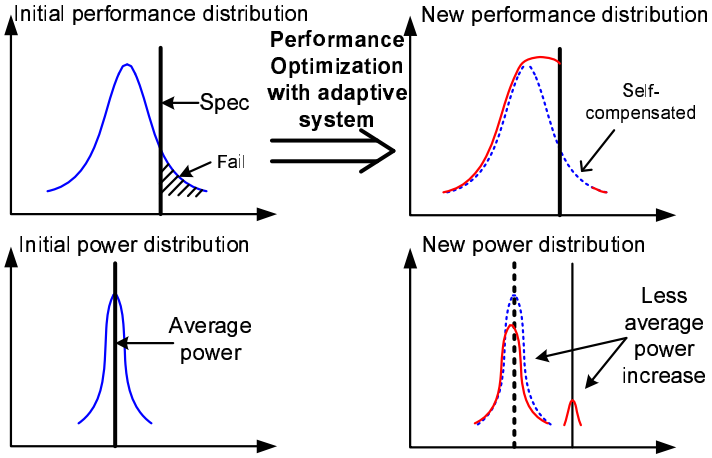

Fig. 57. Proposed yield-aware optimization using adaptive operation.

In the next section we look into the optimization of ADPLL systems by correctly modeling the operations of adaptive function and including these information into the system optimization flow.

F. Optimization of Adaptive ADPLLs

1. Adaptive System Performance Calculation

The optimization of adaptive ADPLLs also depends on the variation-aware block performance models for TDC, DCO and filter. The performance and yield calculation in adaptive systems are different from conventional systems since the adaptation

operation needs to be accounted. The system is reconfigured when the performance monitoring block indicates failure to meet specification for certain process variation scenario. So the calculation of statistical performances in adaptive systems is essential for the system optimization.

The algorithm to find performance distributions under process variations for a certain design parameter set $\overrightarrow{D}$ and topology set $\overrightarrow{T}$ is illustrated in Algorithm 3. The target of the algorithm is to find the jitter and power distribution efficiently, which is also implemented in hardware to perform ADPLL adaptation control. The main idea is to find the system configuration with lowest power consumption while still meets the required jitter specification. The jitter calculation is based on the integration of system noise [66].

As indicated in the algorithm, we first check if the initial configuration can meet the required system specification. Then we adopt the highest performance configuration to see if it can meet the specification. If the chip with configuration $\overrightarrow{C}_{max}$ meets the specifications, we continue to search and save the configuration with the lowest power consumption, otherwise the chip is classified as failure since it can not work in the best performance configuration.

2. Optimization of Adaptive ADPLLs

We can perform different optimization tasks for adaptive ADPLL systems. Here we select the optimization target as to minimize overall system power while keeping the yield at the required level. The statistical power and jitter distributions are calculated using Algorithm 3, and the system optimization problem is formulated as

$$\begin{aligned} &\min \quad power(\overrightarrow{D}) \\ &st.\ \ Yield(\overrightarrow{J}) \geq Yield_{req}, \overrightarrow{T} \in \{\overrightarrow{T'}_{opt}\}, \overrightarrow{C} \in \{\overrightarrow{C}\} \end{aligned} \tag{5.20}$$

Algorithm 3 Statistical Performance Calculation for Adaptive ADPLL

Input: ADPLL configuration set $\overrightarrow{C}$, Monte-Carlo process variable samples $\overrightarrow{V} = \{V_1, V_2, \cdots, V_N\}$, initial configuration set $\overrightarrow{C}_{init}$ after topology selection, jitter specification J_{spec}

Output: Jitter distribution $\overrightarrow{J}$ and power distribution $\overrightarrow{P}$.

set $j \leftarrow 1$, $\overrightarrow{C} \leftarrow \overrightarrow{C}_{init}$.
for $i = 1$ to N **do**
  calculate system jitter $J(\overrightarrow{C}, \overrightarrow{T}, \overrightarrow{D}, V_i)$ under process variable set V_i;
  if $J(\overrightarrow{C}, \overrightarrow{T}, \overrightarrow{D}, V_i) > J_{spec}$ **then**
    select highest performance and power configuration $\overrightarrow{C}_{max}$ for ADPLL
    if $J(\overrightarrow{C}_{max}, \overrightarrow{T}, \overrightarrow{D}, V_i) > J_{spec}$ **then**
      return.
    else
      for configuration $\overrightarrow{C}_k$ increase performance and power from $\overrightarrow{C}_{init}$ to $\overrightarrow{C}_{max}$ **do**
        if $J(\overrightarrow{C}_k, \overrightarrow{T}, \overrightarrow{D}, V_i) > J_{spec}$ **then**
          save $J_j = J(\overrightarrow{C}_k, \overrightarrow{T}, \overrightarrow{D}, V_i)$, $P_j = P(\overrightarrow{C}_k, \overrightarrow{T}, \overrightarrow{D}, V_i)$.
          $j \leftarrow j + 1$
          break.
        end if
      end for
    end if
  else
    save $J_j = J(\overrightarrow{C}, \overrightarrow{T}, \overrightarrow{D}, V_i)$, $P_j = P(\overrightarrow{C}, \overrightarrow{T}, \overrightarrow{D}, V_i)$.
    set $j \leftarrow j + 1$.
  end if
end for

where $\{\overrightarrow{C}\}$ are adaptive tuning space which are determined by the implementation of ADPLL systems.

We perform the ADPLL topology selection similarly as in the conventional ADPLL optimization, then search within the optimized topology sets $\{\overrightarrow{T'}_{opt}\}$ after topology evaluation, note that the filter order in topology selection $\overrightarrow{T}$ is reconfigurable and considered as part in $\overrightarrow{C}$, so we rewrite $\{\overrightarrow{T}_{opt}\}$ as $\{\overrightarrow{T'}_{opt}\}$. The formulation of the optimization problem is in Algorithm 4.

Algorithm 4 Fine Tuning for Adaptive ADPLLs

Input: optimized topology sets $\{\overrightarrow{T'}_{opt}\}$, design variables $\overrightarrow{D}$, initial design $\overrightarrow{D}_{init}$ after topology optimization, required jitter yield $Yield_{req}$
Output: topology and design variables for minimum power.

1: **for** all topologies in $\{\overrightarrow{T'}_{opt}\}$ **do**
2: set $\overrightarrow{D} \leftarrow \overrightarrow{D}_{init}$
3: calculate jitter $\overrightarrow{J}(\overrightarrow{D})$ and power $\overrightarrow{P}(\overrightarrow{D})$ distribution using Algorithm 1, with 200 process variation samples.
4: calculate jitter yield $Yield_J$ and average power $Avg(power)$.
5: **if** $Yield_J < Yield_{req}$ **then**
6: add penalty $p \propto \exp(Yield_{req} - Yield_J)$ to object function Avg(power) for convergence speedup.
7: **end if**
8: **while** min($Avg(power)$) not converge **do**
9: set new $\overrightarrow{D}$ using coordinate search [34]
10: repeat step 3-7
11: **end while**
12: save minimum power of current topology.
13: **end for**
14: select the topology and design variables of minimum power.

G. Experimental Results

In this section, we demonstrate the effectiveness of the proposed yield-aware system optimization approach for ADPLLs by going through a design example. The building block designs are implemented in a $90nm$ CMOS technology with 1.2V power

Table XIII. Optimization variable summary.

		T (selection)	D (variable)
DCO		2 (DAC or SDM)	4
TDC		2 (Reg. or Fine)	4
Filter	Type	2 (I or II)	N/A
	Gain	N/A	1
	IIR order	4 (0 or 1 or 2 or 3)	N/A
	IIR gain	N/A	3

supply. TDC and DCO options are designed manually and the digital circuits are synthesized using standard cell libraries. The performance models of TDC and DCO are generated using Kriging models [10] with data samples collected using SpectreRF [63] simulations. Process variation information is extracted using PDK statistical simulation and included in the system performance models.

The optimization targets are set to the system jitter integrated from 1KHz to 10MHz based on [66], system power and system area. The jitter specification is set to $0.5ps$, the power specification is set to $15mW$ and the area specification is set to $0.5mm^2$.

1. Normal ADPLL Optimization

We first optimize an ADPLL without adaptive tuning function. The numbers of topology choices and the dimensions of TDC and DCO design variables are summarized in Table XIII.

Event-driven simulation technique is of good accuracy and much faster than SPICE-like simulators for the ADPLL system performance evaluation [67]. We compare the phase noise spectrum calculated using the proposed approach and that from

the event-driven simulation in Fig. 58. The close fitting of the two curves demonstrates the accuracy of our proposed transfer function based simulation method.

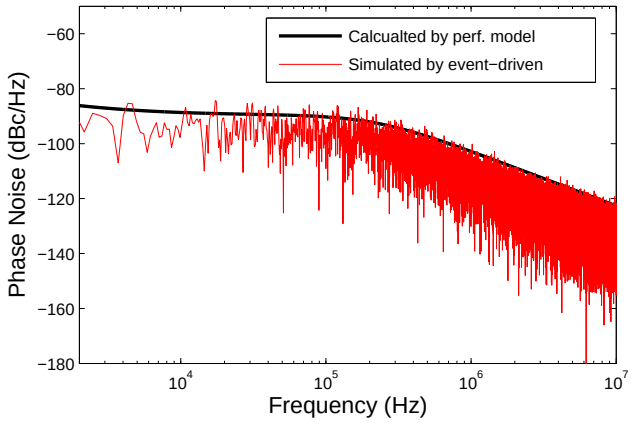

Fig. 58. Comparison of phase noise obtained by proposed method and event-driven simulation.

In the topology selection stage, we plot the jitter distribution with all discretized topology choices in Fig. 59. As can be seen from the figure, the major portion of the discretized system design variables generate much higher jitters than the specification, so we can safely filter out these design points after the topology selection stage and use the remaining topology choices for the transistor-level fine tuning.

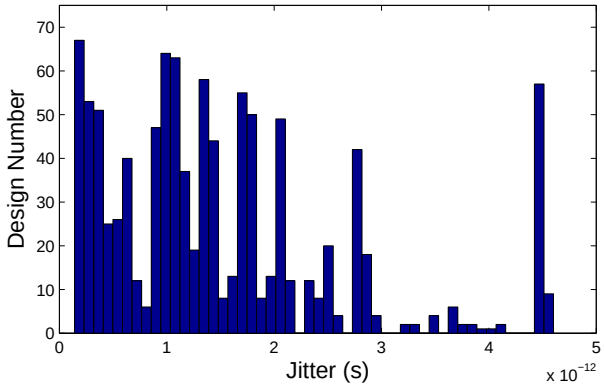

Fig. 59. Jitter distribution in topology selection.

The proposed optimization flow is quite efficient and can be finished within about one hour. In the fine tuning stage, system performance distributions are optimized. The jitter distribution of the initial design is compared with the optimized design after fine tuning in Fig. 60.

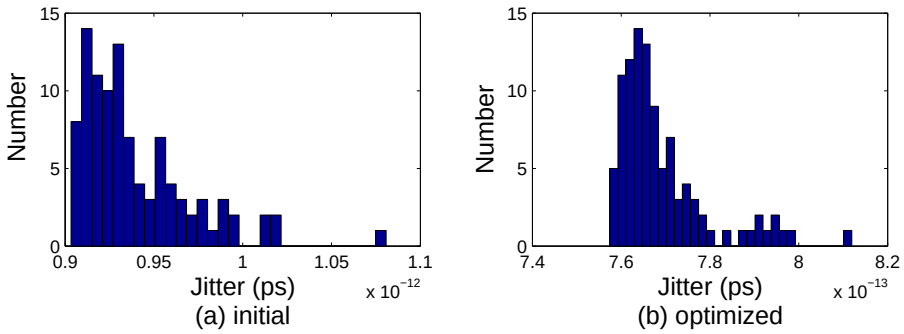

Fig. 60. Jitter distribution comparisons in fine tuning stage.

2. Adaptive ADPLL Optimization

We use the similar setups as the previous example for the optimization of adaptive ADPLL. In the topology selection stage, we evaluate different system topologies and build the idea of the system performance trade-offs and near-optimal circuit configurations. The yield-aware topology evaluation results for power and jitter are plotted in Fig. 61.

Similarly the trade-offs of area and jitter performances are plotted in Fig. 62. From the figure we can see that the area changes are small (less than 10%) and all meet the requried specification. In the ADPLL fine tuning stage, we focus on the power minimization.

We set the yield target at 99% and minimize the power consumption using the approaches presented in Algorithm 4. The power distribution after conventional yield-aware tuning is presented in Fig. 63.

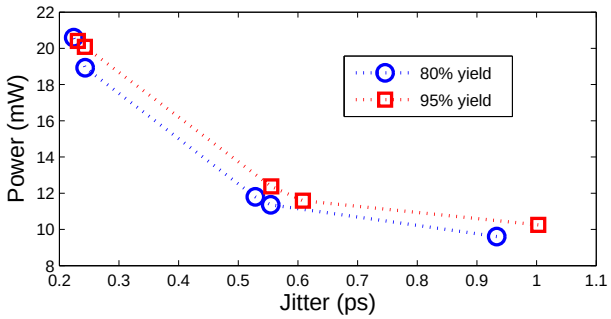

Fig. 61. Power and jitter trade-offs in topology evaluation.

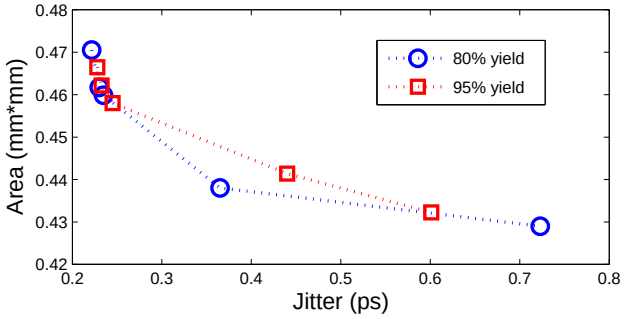

Fig. 62. Area and jitter trade-offs in topology evaluation.

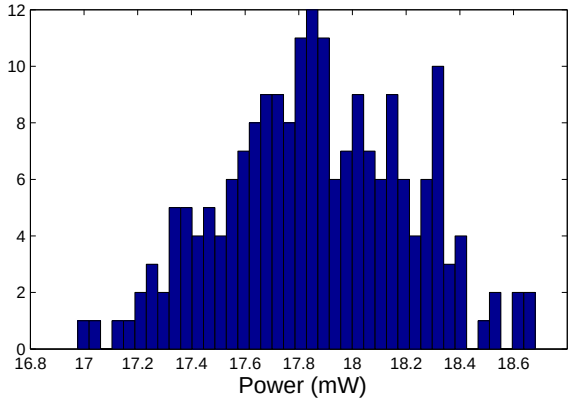

Fig. 63. Power distribution for the reference optimization.

As comparison, the power distribution for the adaptive ADPLL system is shown in Fig. 64. The average power value for the traditional optimization is $17.71mW$ which the one of the adaptive ADPLL is $13.76mW$, which presents 22.3% power reduction.

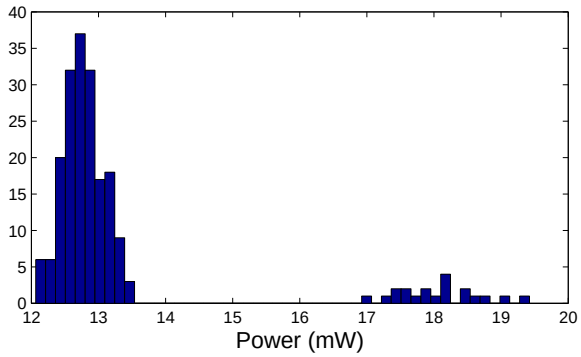

Fig. 64. Power distribution for the adaptive optimization.

H. Summary and Discussion

A mixed-signal circuit design example is presented in this chapter. We design an all-digital PLL circuit to apply the modeling, optimization and testing ideas illustrated in the previous chapters. The noise performances of ADPLL are calculated using the efficient s-domain transfer function approach. Process variation-aware models are developed for the building blocks of analog natures, these models are then utilized to optimize ADPLL system performances. The yield-aware ADPLL optimization is achieved efficiently by first performing topology selection and then doing fine tuning for design variables. System performance self-healing functions are analyzed and implemented to enhance the ADPLL performances. Experimental results demonstrate the effectiveness of our modeling and optimization framework and also indicate that adaptive system designs have the ability to achieve better overall system performances

than these of only performing automatic transistor sizing.

We have demonstrated that the system adaption is very useful in yield enhancements for analog/mixed-signal circuits in scaled CMOS technologies. The failure chips can be brought back to the working condition with on-chip performance detection and system self-healing functions. In addition to the performance enhancements, system adaption can also be used to downgrade the system performances to save power, as illustrated in Fig. 65. Some portion of the fabricated chips may have much higher performances with large power consumptions, which can be considered as overdesign. It is of interest to reduce the power of these chips by "self-downgrading" the system performances and lowing the overall power consumption. This "two-way" style system adaption optimizes system performances and power whenever possible with single on-chip logic function and hence can achieve even better overall system performances.

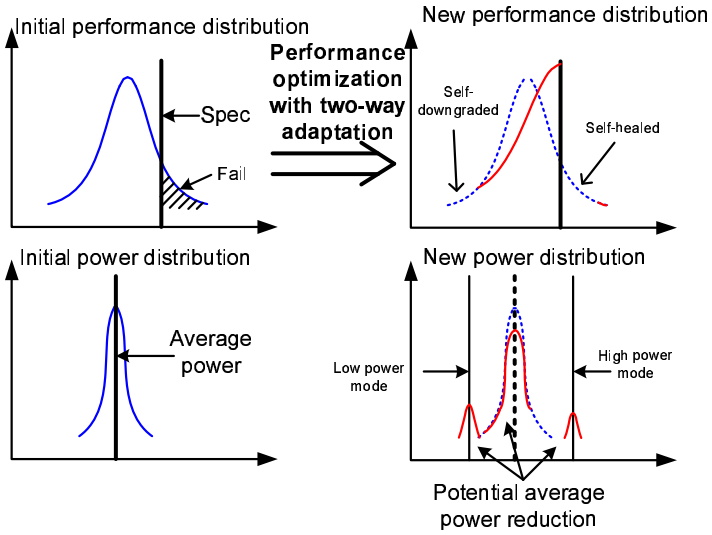

Fig. 65. Yield-aware optimization with two-way adaption.

In real system implementations, the power reduction in the system adaption can be achieved by reconfiguring supply voltage [68] or body bias [69]. In the ADPLL

design case, we can also consider the DCO biasing current as a tunable variable to save the system power. In this dissertation we do not focus on using the system adaption to push power reduction since the circuits are designed to be of the minimum power consumption at the starting point. However, it is still of potential benefit to have both the self-healing and self-downgrading system adaption functions to further enhance the system performances.

CHAPTER VI

CONCLUSIONS AND FUTURE DIRECTIONS

A. Conclusions

The focuses of this dissertation are employing computer methods to design and optimize large analog/mixed-signal systems in highly scaled CMOS technologies. We have addressed the problems of efficient circuit performance modeling and expeditious simulation for robust Sigma-Delta ADC and Phase-locked Loop designs with consideration of process variations. General performance modeling of small scale analog blocks are achieved using Kriging modeling method which is integrated in the framework of yield-aware hierarchical analog system optimization. We have also investigated the problem of robust circuit design by employing built-in self-testing circuitries in analog/mixed-signal systems so that individual chips can perform self-healing of performances after fabrication to fight against process variations. A circuit design example of digital-intensive PLL is presented to illustrated the ideas of large mixed-signal system modeling, optimization and testing.

B. Future Directions

Currently in the industry, most portion of analog circuits are still designed by experienced circuit engineers instead of automatic design approaches. The designers utilize their expertise in circuit designs and the process information from the foundries to accomplish robust analog/mixed-signal designs. The immediate drawbacks of such approaches turn out to be 1) rapidly increasing design closure cycle, the designers need to wait for days or weeks to justify a small design change using current transistor-level simulation tools, 2) the large number of tunable design parameters makes it

almost impossible to achieve global optima by performing handcrafted design, and 3) the inefficient use of foundry process variation information may cost a few tape-out iterations before converging to yield-stable design choices.

Despite of the shortcomings of manual design procedures, automatic circuit design ideas have not been adopted by the majority in analog design society yet, when compared with the grate success of digital VLSI design methodologies. There are many reasons to explain this situation but it clearly indicates there are missing pieces and still long way to go for analog EDA research and developments. The barriers keeping analog circuit designers from adopting automatic design methodologies come from various aspects. The accuracy concerns of fast performance evaluation and optimization tools always stay on the top of circuit designers' minds. Many computer-aided design enhancement tools are circuit topology specified and require dedicated human involvement with steep learning curves, which makes them difficult to build standardized design flows. Human beings is always reluctant to switch from familiar and established procedures to something unfamiliar unless completely necessary. And lastly computer designs can not do innovations which are valued as crucial by some analog designers.

As CMOS technologies scale further following Moore's law (or the development of technologies), there could be increasing opportunities for automatic analog design methodologies to gain popularity. Several scenarios could happen in the next few years/decades and make the automatic design method essential in industry design practice. For example, the process variations might become so severe (refer to Table I) that the impacts of variabilities may change the circuit characters instead of inflecting them in quantities. For high-performance or high-yield circuit designs, circuit layout have to be designed and calibrated by the foundries for individual manufacturing processes to achieve best manufacturability. In that case, the foundries have

to build and verify element circuits and their layouts, then ship the performance and characteristic models of these circuits as fundamental building blocks to the designers/system integrators to design electronic systems. It is no longer necessary for the designers to deal with transistor-level design problems under such circumstances. Clearly the modeling and optimization techniques proposed in this dissertation would be of great value for such kind of digital-like design procedures in the future. The recent acquisition of the world's largest analog IP provider by the EDA giant Synopsys signals that analog design revolution might be already on the way [70].

REFERENCES

[1] *The International Technology Roadmap for Semiconductors*, http://public.itrs.net, Accessed July 2009.

[2] S. Nassif, "Process variability at the 65nm node and beyond," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2008, pp. 1–8.

[3] G. Gielen and R. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proc. of the IEEE*, vol. 88, no. 12, pp. 1825–1852, Dec. 2000.

[4] S. R. Norsworthy, R. Schreier, and G. C. Temes, *Delta-Sigma Data Converters: Theory, Design, and Simulation*, Piscataway, NJ: IEEE Press, 1997.

[5] R. Best, *Phase-Locked Loops Design, Simulation, and Applications 5th edition*, New York City, NY: McGraw-Hill Professional, 2003.

[6] G. Matheron, "Principles of geostatistics," *Economic Geology*, vol. 58, no. 8, pp. 1246–1266, Dec. 1963.

[7] R. Staszewski and P. Balsara, *All-Digital Frequency Synthesizer in Deep-submicron CMOS*, Hoboken, NJ: Wiley-Interscience, 2008.

[8] G. Yu and P. Li, "Lookup table based simulation and statistical modeling of sigma-delta ADCs," in *Proc. IEEE/ACM Design Automation Conf.*, 2006, pp. 1035–1040.

[9] G. Yu and P. Li, "A methodology for systematic built-in self-test of phase-locked loops targeting at parametric failures," in *Proc. of IEEE Int. Test Conf.*, 2007, pp. 1–10.

[10] G. Yu and P. Li, "Yield-aware analog integrated circuit optimization using geostatistics motivated parametric failures," in *Proc. of IEEE/ACM Int. Conf. on CAD*, 2007, pp. 464–469.

[11] Cadence Design Systems Technical Staff, *Affirma Spectre Circuit Simulator User Guide*, Cadence Design Systems, Inc., San Jose, CA, 2000.

[12] R. J. Bishop, J.J. Paulos, M. B. Steer, and S. H. Ardalan, "Table-based modeling of delta-sigma modulators," *IEEE Trans. Circuits Syst.*, vol. 37, no. 3, pp. 447–451, March 1990.

[13] K. K. Low and S. W. Director, "An efficient methodology for building macromodels of IC fabrication processes," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 8, no. 12, pp. 1299–1313, December 1989.

[14] G.E.P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters : Design, Innovation, and Discovery*, Hoboken, NJ: John Wiley & Son, 2005.

[15] J. Zou, D. Mueller, H. Graeb, and U. Schlichtmann, "A CPPLL hierarchical optimization methodology considering jitter, power and locking time," in *Proc. of IEEE/ACM Design Automation Conf*, 2006, pp. 19–24.

[16] N. Godambe and C. J. R. Shi, "Behavioral level noise modeling and jitter simulation of phase-locked loops with faults using VHDL-AMS," in *Proc. of IEEE VLSI Test Symposium*, 1997, pp. 177–182.

[17] A. Phanse, R. Shirani, R. Rasmussen, R. Mendel and J. Yuan, "Behavioral modeling of a phase locked loop," in *Southcon'96*, 1996, pp. 400–404.

[18] S. R. Nassif, "Modeling and analysis of manufacturing variations," in *IEEE Custom Integrated Circuits Conference*, 2001, pp. 223–228.

[19] D. Morrison, *Multivariate Statistical Methods*, New York City, NY: McGraw-Hill, 1976.

[20] Z. Feng and P. Li, "Performance-oriented statistical parameter reduction of parameterized systems via reduced rank regression," in *Proc. of IEEE/ACM Int. Conf. on CAD*, 2006, pp. 868–875.

[21] G. Reinsel and R. Velu, *Multivariate Reduced-Rank Regression, Theory and Applications*, New York City, NY: Springer-Verlag, 1998.

[22] J. Sacks, W. Welch, T. Mitchell, and H. Wynn, "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–435, Nov. 1989.

[23] M. Bernardo, R. Buck, L. Liu, W. Nazaret, J. Sacks, and W. Welch, "Integrated circuit design optimization using a sequential strategy," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 3, pp. 361–372, March 1992.

[24] G. Yu and P. Li, "Yield-aware hierarchical optimization of large analog integrated circuits," in *Proc. of IEEE/ACM Int. Conf. on CAD*, 2008, pp. 79–84.

[25] M. Driscoll, W. Daasch, and C. Sembakutti, "Efficient design centering of analog integrated circuits using binary search," *Analog Integrated Circuits and Signal Processing*, vol. 6, pp. 157 – 169, 1994.

[26] F. Schenkel, M. Pronath, S. Zizala, R. Schwencker, H. Graeb, and K. Antreich, "Mismatch analysis and direct yield optimization by specwise linearization and feasibility-guided search," in *Proc. of IEEE Design Automation Conference*, 2001, pp. 858–863.

[27] Y. Xu, K. Hsiung, X. Li, I. Vausieda, S. Boyd and L. Pileggi, "OPERA: Optimization with ellipsoidal uncertainty for robust analog IC design," in *Proc. of*

IEEE/ACM Design Automation Conference, 2005, pp. 632–637.

[28] Cadence Design Systems Technical Staff, *Virtuoso NeoCircuit User Guide*, Cadence Design Systems, 2009.

[29] MunEDA Technical Staff, *WiCkeD User Guide*, MunEDA, 2009.

[30] K. Antreich, H. Graeb, and C. Wieser, "Circuit analysis and optimization driven by worst-case distancess," *IEEE Trans. on Computer-aided Design*, vol. 13, no. 1, pp. 57 – 71, Jan. 1994.

[31] G. Stehr, H. Braeb, and K. Antreich, "Performance trade-off analysis of analog circuits by normal-boundary intersection," in *Proc. of IEEE/ACM Design Automation Conference*, 2003, pp. 958–963.

[32] S. Tiwary, P. Tiwary, and R. Rutenbar, "Generation of yield-aware pareto surfaces for hierarchical circuit design space exploration," in *Proc. of IEEE/ACM Design Automation Conf*, 2006, pp. 31–36.

[33] B. Smedt and G. Gielen, "HOLMES: Capturing the yield - optimized design space boundaries of analog and RF integrated circuits," in *Proc. of IEEE/ACM Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 19–24.

[34] W. Huyer and A. Neumaier, "Global optimization by multilevel coordinate search," *Journal of Global Optimization*, vol. 14, no. 4, pp. 331–355, June 1999.

[35] S. Tiwary, S. Velu, R. Butenbar, and T. Mukherjee, "Pareto optimal modeling for efficient PLL optimization," in *Nanotech 2004 Vol. 2*, 2004, pp. 195–198.

[36] Open Verilog International, *Verilog-A Language Reference Manual*, www.verilog.org, Accessed Sept. 2009.

[37] K. Kundert, *Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers*, www.designers-guide.org, Accessed Aug. 2006.

[38] G. Roberts, "Metrics, techniques and recent developments in mixed-signal testing," in *Proc. of IEEE/ACM Int. Conf. on CAD*, 1996, pp. 514 – 521.

[39] G. Yu, P. Li, and W. Dong, "Achieving low-cost linearity test and diagnosis of sigma delta ADCs via frequency-domain nonlinear analysis and macromodeling," in *Proc. of IEEE Int. Symposium on Quality Electronic Design*, 2007, pp. 513–518.

[40] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*, Hoboken, NJ: John Wiley & Son, 1980.

[41] P. Wambacq and W. Sansen, *Distortion Analysis of Analog Integrated Circuits*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998.

[42] W. J. Rugh, *Nonlinear System Theory - The Volterra/Wiener Approach*, Baltimore, MD: The Johns Hopkins University Press, 1981.

[43] N. Csizmadia and A.J.E.M. Janssen, "Estimating the integral non-linearity of AD-converters via the frequency domain," in *IEEE Int. Test Conf.*, 1999, pp. 757–762.

[44] *INL/DNL measurements for high-speed analog-to-digital converters ADCs - AN 283*, Maxim Integrated Products, Dallas, TX, Nov. 2001.

[45] V. Vapnik, *Statistical Learning Theory*, Hoboken, NJ: Wiley-Interscience Publishers, 1998.

[46] S. Sunter and A. Roy, "BIST for phase-locked loops in digital applications," in *Proc. of IEEE International Test Conference*, 1999, pp. 532–540.

[47] S. Kim and M. Soma, "An all-digital built-in self-test for high-speed phase-locked loops," *IEEE Trans. on Circuit and Systems -II: Analog and Digital Signal Processing*, vol. 48, no. 2, pp. 141–150, Feb. 2001.

[48] C. Hsu, Y. Lai and S. Wang, "Built-in self-test for phase-locked loops," *IEEE Trans. on Instr. and Measurement*, vol. 54, no. 3, pp. 996–1002, June 2005.

[49] F. Azais, Y. Bertrand, M. Renovell, A. Ivanov and S. Tabatabaei, "An all-digital DFT scheme for testing catastrophic faults in PLLs," *IEEE Design & Test of Computers*, vol. 20, no. 1, pp. 60–67, Jan. 2003.

[50] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Statistical modeling of device mismatch for analog integrated circuits," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1440, October 1989.

[51] U. Schaper, J. Einfeld, and A. Sauerbrey, "Parameter variation on chip-level," in *IEEE Int. Conf. on Microelectronic Test Structures*, 2005, pp. 155–158.

[52] J. Dunning, G. Garcia, J. Lundberg, and E. Nuckolls, "An all-digital phase-locked loop with 50-cycle lock time suitable for high-performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 30, no. 4, pp. 412–422, Apr. 1995.

[53] C. Chung and C. Lee, "An all-digital phase-locked loop for high-speed clock generation," *IEEE J. Solid-State Circuits*, vol. 38, no. 2, pp. 347–351, Feb. 2003.

[54] R. Staszewski, K. Muhammad, D. Leipold, C. Hung, Y. Ho, and et al, "All-digital TX frequency synthesizer and discrete-time receiver for bluetooth radio in 130-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 39, no. 12, pp. 2278–2291, Dec. 2004.

[55] H. Chang, P. Wang, J. Zhan, and B. Hsieh, "A fractional spur-free ADPLL with loop-gain calibration and phase-noise cancellation for GSM/GPRS/EDGE," in *Proc. of ISSCC Dig. Tech. Papers*, 2008, pp. 200–202.

[56] C. Hsu, M. Straayer, and M. Perrott, "A low-noise wide-BW 3.6GHz digital ds fractional-N frequency synthesizer with a noise-shaping time-to-digital converter and quantization noise cancellation," in *Proc. of ISSCC Dig. Tech. Papers*, 2008, pp. 340–342.

[57] C. Wu, E. Temporiti, D. Baldi, and F. Svelto, "A 3GHz fractional-N all-digital PLL with precise time-to-digital converter calibration and mismatch correction," in *Proc. of ISSCC Dig. Tech. Papers*, 2008, pp. 344–346.

[58] R. Staszewski, S. Vemulapalli, P. Vallur, J. Wallberg, and P.T. Balsara, "1.3 V 20 ps time-to-digital converter for frequency synthesis in 90-nm CMOS," *IEEE Tran. on Circuits and Systems. - II*, vol. 53, no. 4, pp. 769–777, Mar. 2006.

[59] C. Lau and M. Perrott, "Fractional-N frequency synthesizer design at the transfer function level using a direct closed loop realization algorithm," in *Proc. of IEEE/ACM Design Automation Conference*, July 2003, pp. 526–531.

[60] C. Hsu, *Techniques for High-Performance Digital Frequency Synthesis and Phase Control*, PhD dissertation, MIT, Cambridge, MA, 2008.

[61] F. Gardner, "Charge-pump phase-locked loops," *IEEE Trans. on Communications*, vol. 28, pp. 1849 – 1858, Nov. 1980.

[62] M. Lee and A. Abidi, "A 9b, 1.25 ps resolution coarse-fine time-to-digital converter in 90 nm CMOS that amplifies a time residue," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 769–777, Apr. 2008.

[63] Cadence Design Systems Technical Staff, *SpectreRF Circuit Simulator User Guide*, Cadence Design Systems, June 2008.

[64] K. Waheed and R. Staszewski, "Digital RF processing techniques for device mismatch tolerant transmitters in nanometer-scale CMOS," in *Proc. of IEEE International Symposium on Circuits and Systems*, May 2007, pp. 1253–1256.

[65] R. Staszewski, I. Bashir, and O. Eliezer, "RF built-in self test of a wireless transmitter," *IEEE Tran. on Circuits and Systems - II*, vol. 39, no. 12, pp. 2278–2291, Dec. 2004.

[66] *Clock Jitter and Phase Noise Conversion - Maxim AN 3359*, Maxim Integrated Products, Dallas, TX, Dec. 2004.

[67] R. Staszewski, C. Fernando, and P. Balsara, "Event-driven simulation and modeling of phase noise of an RF oscillator," *IEEE Tran. on Circuits and Systems. - I*, vol. 52, no. 4, pp. 723–733, Apr. 2005.

[68] T. Chen and S. Naffziger, "Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation," *IEEE Trans. on VLSI*, vol. 11, no. 5, pp. 888–899, May 2003.

[69] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE J. on Solid-States Circuits*, vol. 27, no. 11, pp. 1396–1402, Nov. 2002.

[70] *Synopsys Acquires Analog Business Group of MIPS Technologies*, PR Newswire, Mountain View, CA, May 2009.

VITA

Guo Yu received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China and Delft University of Technology (TU Delft), Delft, the Netherlands, in 2003 and 2005, respectively. He worked at Philips Semiconductors Netherlands from Oct. 2004 to Aug. 2005, and Cadence Design Systems Pittsburgh during the summer of 2007 as a research intern respectively. His research interests include modeling and optimization for analog/mixed-signal circuits, built-in test scheme design, timing and yield analysis for VLSI circuits, and device modeling. He can be reached at WERC 331A, Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3128.

The typist for this thesis was Guo Yu.